Least Squares Data Fitting

WITH APPLICATIONS

Per Christian Hansen
Víctor Pereyra
Godela Scherer

VISIT…

Least Squares Data Fitting with Applications

This page intentionally left blank

Least Squares Data Fitting with Applications

PER CHRISTIAN HANSEN
VÍCTOR PEREYRA
GODELA SCHERER

The Johns Hopkins University Press
Baltimore

© 2013 The Johns Hopkins University Press
All rights reserved. Published 2013
Printed in the United States of America on acid-free paper
9 8 7 6 5 4 3 2 1

The Johns Hopkins University Press
2715 North Charles Street
Baltimore, Maryland 21218-4363
www.press.jhu.edu

ISBN-13: 978-1-4214-0786-9 (hardcover : alk. paper)
ISBN-10: 1-4214-0786-8 (hardcover : alk. paper)
ISBN-13: 978-1-4214-0858-3 (electronic)
IBSN-10: 1-4214-0858-9 (electronic)

Library of Congress Control Number: 2012942511

A catalog record for this book is available from the British Library.

Special discounts are available for bulk purchases of this book. For more information, please contact Special Sales at 410-516-6936 or specialsales@press.jhu.edu.

The Johns Hopkins University Press uses environmentally friendly book materials, including recycled text paper that is composed of at least 30 percent post-consumer waste, whenever possible.

Contents

Preface **ix**

Symbols and Acronyms **xiii**

1 The Linear Data Fitting Problem **1**
1.1 Parameter estimation, data approximation 1
1.2 Formulation of the data fitting problem 4
1.3 Maximum likelihood estimation 9
1.4 The residuals and their properties 13
1.5 Robust regression 19

2 The Linear Least Squares Problem **25**
2.1 Linear least squares problem formulation 25
2.2 The QR factorization and its role 33
2.3 Permuted QR factorization 39

3 Analysis of Least Squares Problems **47**
3.1 The pseudoinverse 47
3.2 The singular value decomposition 50
3.3 Generalized singular value decomposition 54
3.4 Condition number and column scaling 55
3.5 Perturbation analysis 58

4 Direct Methods for Full-Rank Problems **65**
4.1 Normal equations 65
4.2 LU factorization 68
4.3 QR factorization 70
4.4 Modifying least squares problems 80
4.5 Iterative refinement 85
4.6 Stability and condition number estimation 88
4.7 Comparison of the methods 89

5 Direct Methods for Rank-Deficient Problems 91
5.1 Numerical rank 92
5.2 Peters-Wilkinson LU factorization 93
5.3 QR factorization with column permutations 94
5.4 UTV and VSV decompositions 98
5.5 Bidiagonalization 99
5.6 SVD computations 101

6 Methods for Large-Scale Problems 105
6.1 Iterative versus direct methods 105
6.2 Classical stationary methods 107
6.3 Non-stationary methods, Krylov methods 108
6.4 Practicalities: preconditioning and stopping criteria 114
6.5 Block methods 117

7 Additional Topics in Least Squares 121
7.1 Constrained linear least squares problems 121
7.2 Missing data problems 131
7.3 Total least squares (TLS) 136
7.4 Convex optimization 143
7.5 Compressed sensing 144

8 Nonlinear Least Squares Problems 147
8.1 Introduction 147
8.2 Unconstrained problems 150
8.3 Optimality conditions for constrained problems 156
8.4 Separable nonlinear least squares problems 158
8.5 Multiobjective optimization 160

9 Algorithms for Solving Nonlinear LSQ Problems 163
9.1 Newton's method 164
9.2 The Gauss-Newton method 166
9.3 The Levenberg-Marquardt method 170
9.4 Additional considerations and software 176
9.5 Iteratively reweighted LSQ algorithms for robust data fitting problems 178
9.6 Variable projection algorithm 181
9.7 Block methods for large-scale problems 186

10 Ill-Conditioned Problems **191**
10.1 Characterization 191
10.2 Regularization methods 192
10.3 Parameter selection techniques 195
10.4 Extensions of Tikhonov regularization 198
10.5 Ill-conditioned NLLSQ problems 201

11 Linear Least Squares Applications **203**
11.1 Splines in approximation 203
11.2 Global temperatures data fitting 212
11.3 Geological surface modeling 221

12 Nonlinear Least Squares Applications **231**
12.1 Neural networks training 231
12.2 Response surfaces, surrogates or proxies 238
12.3 Optimal design of a supersonic aircraft 241
12.4 NMR spectroscopy 248
12.5 Piezoelectric crystal identification 251
12.6 Travel time inversion of seismic data 258

Appendix A Sensitivity Analysis **263**
A.1 Floating-point arithmetic 263
A.2 Stability, conditioning and accuracy 264

Appendix B Linear Algebra Background **267**
B.1 Norms 267
B.2 Condition number 268
B.3 Orthogonality 269
B.4 Some additional matrix properties 270

Appendix C Advanced Calculus Background **271**
C.1 Convergence rates 271
C.2 Multivariable calculus 272

Appendix D Statistics **275**
D.1 Definitions 275
D.2 Hypothesis testing 280

References **281**

Index **301**

This page intentionally left blank

Preface

This book surveys basic modern techniques for the numerical solution of linear and nonlinear least squares problems and introduces the treatment of large and ill-conditioned problems. The theory is extensively illustrated with examples from engineering, environmental sciences, geophysics and other application areas.

In addition to the treatment of the numerical aspects of least squares problems, we introduce some important topics from the area of regression analysis in statistics, which can help to motivate, understand and evaluate the computed least squares solutions. The inclusion of these topics is one aspect that distinguishes the present book from other books on the subject.

The presentation of the material is designed to give an overview, with the goal of helping the reader decide which method would be appropriate for a given problem, point toward available algorithms/software and, if necessary, help in modifying the available tools to adapt for a given application. The emphasis is therefore on the properties of the different algorithms and few proofs are presented; the reader is instead referred to the appropriate articles/books. Unfortunately, several important topics had to be left out, among them, direct methods for sparse problems.

The content is geared toward scientists and engineers who must analyze and solve least squares problems in their fields. It can be used as course material for an advanced undergraduate or graduate course in the sciences and engineering, presupposing a working knowledge of linear algebra and basic statistics. It is written mostly in a terse style in order to provide a quick introduction to the subject, while treating some of the not so well-known topics in more depth. This in fact presents the reader with an opportunity to verify the understanding of the material by completing or providing the proofs without checking the references.

The least squares problem is known under different names in different disciplines. One of our aims is to help bridge the communication gap between the statistics and the numerical analysis literature on the subject, often due to the use of different terminology, such as l_2-approximation,

regularization, regression analysis, parameter estimation, filtering, process identification, etc.

Least squares methods have been with us for many years, since Gauss invented and used them in his surveying activities [83]. In 1965, the paper by G. H. Golub [92] on using the QR factorization and later his development of a stable algorithm for the computation of the SVD started a renewed interest in the subject in the, by then, changed work environment of computers.

Thanks also to, among many others, Å. Björck, L. Eldén, C. C. Paige, M. A. Saunders, G. W. Stewart, S. van Huffel and P.-. Wedin, the topic is now available in a robust, algorithmic and well-founded form.

There are many books partially or completely dedicated to linear and nonlinear least squares. The first and one of the fundamental references for linear problems is Lawson and Hanson's monograph [150]. Besides summarizing the state of the art at the time of its publication, it highlighted the practical aspects of solving least squares problems. Bates and Watts [9] have an early comprehensive book focused on the nonlinear least squares problem with a strong statistical approach. Björck's book [20] contains a very careful and comprehensive survey of numerical methods for both linear and nonlinear problems, including the treatment of large, sparse problems. Golub and Van Loan's *Matrix Computations* [105] includes several chapters on different aspects of least squares solution and on total least squares. The total least squares problem, known in statistics as latent root regression, is discussed in the book by S. van Huffel and J. Vandewalle [239]. Seber and Wild [223] consider exhaustively all aspects of nonlinear least squares estimation and modeling. Although it is a general treatise on optimization, Nocedal and Wright's book [170] includes a very clear chapter on nonlinear least squares. Additional material can be found in [21, 63, 128, 232, 242, 251].

We would like to acknowledge the help of Michael Saunders (iCME, Stanford University), who read carefully the whole manuscript and made a myriad of observations and corrections that have greatly improved the final product.

Per Christian Hansen would like to thank several colleagues from DTU Informatics who assisted with the statistical aspects.

Godela Scherer gives thanks for all the support at the Department of Mathematics and Statistics, University of Reading, where she was a visiting research fellow while working on this book. In particular, she would like to thank Professor Mike J. Baines and Dr. I Llatas for numerous inspiring discussions.

Victor Pereyra acknowledges Weidlinger Associates Inc. and most especially David Vaughan and Howard Levine, for their unflagging support

and for letting him keep his office and access to computing facilities after retirement. Also Ms. P. Tennant helped immensely by improving many of the figures.

Special thanks are due to the professional handling of the manuscript by the publishers and more specifically to executive editor Vincent J. Burke and production editor Andre Barnett.

Prior to his untimely death on November 2007, Professor Gene Golub had been an integral part of this project team. Although the book has changed significantly since then, it has greatly benefited from his insight and knowledge. He was an inspiring mentor and great friend, and we miss him dearly.

This page intentionally left blank

Symbols and Acronyms

Symbol	Represents
A	$m \times n$ matrix
$A^\dagger, A^-, A^T$	pseudoinverse, generalized inverse and transpose of A
$\boldsymbol{b}$	right-hand side, length m
$\mathrm{cond}(\cdot)$	condition number of matrix in l_2-norm
$\mathrm{Cov}(\cdot)$	covariance matrix
$\mathrm{diag}(\cdot)$	diagonal matrix
$\boldsymbol{e}$	vector of noise, length m
e_i	noise component in data
$\widehat{\boldsymbol{e}}_i$	canonical unit vector
ε_M	machine precision
$\mathcal{E}(\cdot)$	expected value
$f_j(t)$	model basis function
$\Gamma(t)$	pure-data function
$M(\boldsymbol{x}, t)$	fitting model
$\mathcal{N}$	normal (or Gaussian) distribution
$\mathrm{null}(A)$	null space of A
p	degree of polynomial
$P,\ P_i,\ P_{\boldsymbol{x}}$	probability
$\mathcal{P}_X$	projection onto space X
Π	permutation matrix
Q	$m \times m$ orthogonal matrix, partitioned as $Q = (\ Q_1\ Q_2\ )$
$r = r(A)$	rank of matrix A
$\boldsymbol{r},\ \boldsymbol{r}^*$	residual vector, least squares residual vector
r_i	residual for ith data
$\mathrm{range}(A)$	range of matrix A
R, R_1	$m \times n$ and $n \times n$ upper triangular matrix
$\mathrm{span}\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_p\}$	subspace generated by the vectors
Σ	diagonal SVD matrix

Symbol	Represents
ζ, ζ_i	standard deviation
σ_i	singular value
t	independent variable in data fitting problem
t_i	abscissa in data fitting problem
U, V	$m \times m$ and $n \times n$ left and right SVD matrices
$\boldsymbol{u}_i, \boldsymbol{v}_i$	left and right singular vectors, length m and n respectively
W	$m \times m$ diagonal weight matrix
w_i	weight in weighted least squares problem
$\boldsymbol{x}, \boldsymbol{x}^*$	vector of unknowns, least squares solution, length n
$\widehat{\boldsymbol{x}}^*$	minimum-norm LSQ solution
$\boldsymbol{x}_B, \boldsymbol{x}_{\text{TLS}}$	basic LSQ solution and total least squares solution
x_i	coefficient in a linear fitting model
$\boldsymbol{y}$	vector of data in a data fitting problem, length m
y_i	data in data fitting problem
$\|\cdot\|_2$	2-norm, $\|\boldsymbol{x}\|_2 = (x_1^2 + \cdots + x_n^2)^{1/2}$
$\widetilde{\square}$	perturbed version of $\square$

Acronym	**Name**
CG	conjugate gradient
CGLS	conjugate gradient for LSQ
FG	fast Givens
GCV	generalized cross validation
G-N	Gauss-Newton
GS	Gram-Schmidt factorization
GSVD	generalized singular value decomposition
LASVD	SVD for large, sparse matrices
L-M	Levenberg-Marquardt
LP	linear prediction
LSE	equality constrained LSQ
LSI	inequality constrained LSQ
LSQI	quadratically constrained LSQ
LSQ	least squares
LSQR	Paige-Saunders algorithm
LU	LU factorization
MGS	modified Gram-Schmidt
NLLSQ	nonlinear least squares
NMR	nuclear magnetic resonance
NN	neural network
QR	QR decomposition
RMS	root mean square
RRQR	rank revealing QR decomposition
SVD	singular value decomposition
TLS	total least squares problem
TSVD	truncated singular value decomposition
UTV	UTV decomposition
VARPRO	variable projection algorithm, Netlib version
VP	variable projection

This page intentionally left blank

Least Squares Data Fitting with Applications

This page intentionally left blank

Chapter 1

The Linear Data Fitting Problem

This chapter gives an introduction to the linear data fitting problem: how it is defined, its mathematical aspects and how it is analyzed. We also give important statistical background that provides insight into the data fitting problem. Anyone with more interest in the subject is encouraged to consult the pedagogical expositions by Bevington [13], Rust [213], Strutz [233] and van den Bos [242].

We start with a couple of simple examples that introduce the basic concepts of data fitting. Then we move on to a more formal definition, and we discuss some statistical aspects. Throughout the first chapters of this book we will return to these data fitting problems in order to illustrate the ensemble of numerical methods and techniques available to solve them.

1.1 Parameter estimation, data approximation

Example 1. ***Parameter estimation.*** *In food-quality analysis, the amount and mobility of water in meat has been shown to affect quality attributes like appearance, texture and storage stability. The water contents can be measured by means of nuclear magnetic resonance (NMR) techniques, in which the measured signal reflects the amount and properties of different types of water environments in the meat. Here we consider a simplified example involving frozen cod, where the ideal time signal $\phi(t)$ from NMR is a sum of two damped exponentials plus a constant background,*

$$\phi(t) = x_1\, e^{-\lambda_1 t} + x_2\, e^{-\lambda_2 t} + x_3, \qquad \lambda_1, \lambda_2 > 0.$$

In this example we assume that we know the parameters λ_1 and λ_2 that control the decay of the two exponential components. In practice we do not

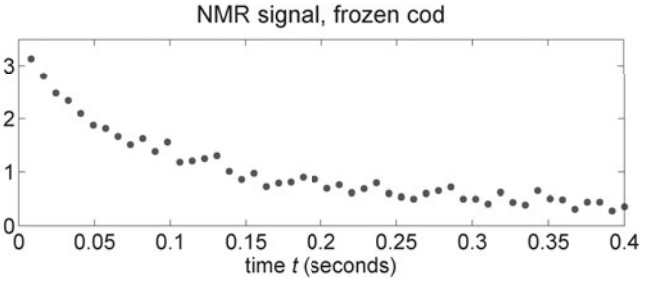

Figure 1.1.1: Noisy measurements of the time signal $\phi(t)$ from NMR, for the example with frozen cod meat.

measure this pure signal, but rather a noisy realization of it as shown in Figure 1.1.1.

The parameters $\lambda_1 = 27\,\mathrm{s}^{-1}$ *and* $\lambda_2 = 8\,\mathrm{s}^{-1}$ *characterize two different types of proton environments, responsible for two different water mobilities. The amplitudes* x_1 *and* x_2 *are proportional to the amount of water contained in the two kinds of proton environments. The constant* x_3 *accounts for an undesired background (bias) in the measurements. Thus, there are three unknown parameters in this model, namely,* x_1, x_2 *and* x_3. *The goal of data fitting in relation to this problem is to use the measured data to estimate the three unknown parameters and then compute the different kinds of water contents in the meat sample. The actual fit is presented in Figure 1.2.1.*

In this example we used the technique of data fitting for the purpose of estimating unknown parameters in a mathematical model from measured data. The model was dictated by the physical or other laws that describe the data.

Example 2. ***Data approximation.*** *We are given measurements of air pollution, in the form of the concentration of NO, over a period of 24 hours, on a busy street in a major city. Since the NO concentration is mainly due to the cars, it has maximum values in the morning and in the afternoon, when the traffic is most intense. The data is shown in Table 1.1 and the plot in Figure 1.2.2.*

For further analysis of the air pollution we need to fit a smooth curve to the measurements, so that we can compute the concentration at an arbitrary time between 0 and 24 hours. For example, we can use a low-degree polynomial to model the data, i.e., we assume that the NO concentration can be approximated by

$$f(t) = x_1\, t^p + x_2\, t^{p-1} + \cdots + x_p\, t + x_{p+1},$$

t_i	y_i	t_i	y_i	t_i	y_i	t_i	y_i	t_i	y_i
0	110.49	5	29.37	10	294.75	15	245.04	20	216.73
1	73.72	6	74.74	11	253.78	16	286.74	21	185.78
2	23.39	7	117.02	12	250.48	17	304.78	22	171.19
3	17.11	8	298.04	13	239.48	18	288.76	23	171.73
4	20.31	9	348.13	14	236.52	19	247.11	24	164.05

Table 1.1: Measurements of NO concentration y_i as a function of time t_i. The units of y_i and t_i are μg/m^3 and hours, respectively.

where t is the time, p is the degree of the polynomial and $x_1, x_2, \ldots, x_{p+1}$ are the unknown coefficients in the polynomial. A better model however, since the data repeats every day*, would use periodic functions:*

$$f(t) = x_1 + x_2 \sin(\omega\, t) + x_3 \cos(\omega\, t) + x_4 \sin(2\omega\, t) + x_5 \cos(2\omega\, t) + \cdots$$

where $\omega = 2\pi/24$ is the period. Again, $x_1, x_2, \ldots$ are the unknown coefficients. The goal of data fitting in relation to this problem is to estimate the coefficients $x_1, x_2, \ldots$, such that we can evaluate the function $f(t)$ for any argument t. At the same time we want to suppress the influence of errors present in the data.

> *In this example we used the technique of data fitting for the purpose of approximating measured discrete data: we fitted a model to given data in order to be able to compute smoothed data for any value of the independent variable in the model. We were free to choose the model as long as it gave an adequate fit to the data.*

Both examples illustrate that we are given data with measurement errors and that we want to fit a model to these data that captures the "overall behavior" of it without being too sensitive to the errors. The difference between the two examples is that in the first case the model arises from a physical theory, while in the second there is an arbitrary continuous approximation to a set of discrete data.

Data fitting is distinctly different from the problem of interpolation, where we seek a model – a function $f(t)$ – that interpolates the given data, i.e., it satisfies $f(t_i) = y_i$ for all the data points. We are not interested in interpolation (which is not suited for noisy data) – rather, we want to approximate the noisy data with a parametric model that is either given or that we can choose, in such a way that the result is not too sensitive to the noise. In this data fitting approach there is redundant data: i.e., more data than unknown parameters, which also helps to decrease the uncertainty in

the parameters of the model. See Example 15 in the next chapter for a justification of this.

1.2 Formulation of the data fitting problem

Let us now give a precise definition of the data fitting problem. We assume that we are given m *data points*

$$(t_1, y_1),\ (t_2, y_2), \ldots,\ (t_m, y_m),$$

which can be described by the relation

$$y_i = \Gamma(t_i) + e_i, \qquad i = 1, 2, \ldots, m. \tag{1.2.1}$$

The function $\Gamma(t)$, which we call the *pure-data function*, describes the noise-free data (it may be unknown, or given by the application), while $e_1, e_2, \ldots, e_m$ are the data errors (they are unknown, but we may have some statistical information about them). The data errors – also referred to as noise – represent measurement errors as well as random variations in the physical process that generates the data. Without loss of generality we can assume that the abscissas t_i appear in non-decreasing order, i.e., $t_1 \leq t_2 \leq \cdots \leq t_m$.

In data fitting we wish to compute an approximation to $\Gamma(t)$ – typically in the interval $[t_1, t_m]$. The approximation is given by the *fitting model* $M(\boldsymbol{x}, t)$, where the vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ contains n parameters that characterize the model and are to be determined from the given noisy data. In the linear data fitting problem we always have a model of the form

$$\boxed{\text{Linear fitting model:} \qquad M(\boldsymbol{x}, t) = \sum_{j=1}^{n} x_j\, f_j(t).} \tag{1.2.2}$$

The functions $f_j(t)$ are called the *model basis functions,* and the number n – the *order* of the fit – should preferably be smaller than the number m of data points. A notable modern exception is related to the so-called compressed sensing, which we discuss briefly in Section 7.5.

The form of the function $M(\boldsymbol{x}, t)$ – i.e., the choice of basis functions – depends on the precise goal of the data fitting. These functions may be given by the underlying mathematical model that describes the data – in which case $M(\boldsymbol{x}, t)$ is often equal to, or an approximation to, the pure-data function $\Gamma(t)$ – or the basis functions may be chosen arbitrarily among all functions that give the desired approximation and allow for stable numerical computations.

The method of least squares (LSQ) is a standard technique for determining the unknown parameters in the fitting model. The *least squares fit*

is defined as follows. We introduce the residual r_i associated with the data points as

$$r_i = y_i - M(\boldsymbol{x}, t_i), \qquad i = 1, 2, \ldots, m,$$

and we note that each residual is a function of the parameter vector $\boldsymbol{x}$, i.e., $r_i = r_i(\boldsymbol{x})$. A least squares fit is a choice of the parameter vector $\boldsymbol{x}$ that minimizes the sum-of-squares of the residuals:

$$\boxed{\text{LSQ fit:} \quad \min_{\boldsymbol{x}} \sum_{i=1}^{m} r_i(\boldsymbol{x})^2 = \min_{\boldsymbol{x}} \sum_{i=1}^{m} \big(y_i - M(\boldsymbol{x}, t_i)\big)^2.} \tag{1.2.3}$$

In the next chapter we shall describe in which circumstances the least squares fit is unique, and in the following chapters we shall describe a number of efficient computational methods for obtaining the least squares parameter vector $\boldsymbol{x}$.

We note in passing that there are other related criteria used in data fitting; for example, one could replace the sum-of-squares in (1.2.3) with the sum-of-absolute-values:

$$\min_{\boldsymbol{x}} \sum_{i=1}^{m} |r_i(\boldsymbol{x})| = \min_{\boldsymbol{x}} \sum_{i=1}^{m} \big|y_i - M(\boldsymbol{x}, t_i)\big|. \tag{1.2.4}$$

Below we shall use a statistical perspective to describe when these two choices are appropriate. However, we emphasize that the book focuses on the least squares fit.

In order to obtain a better understanding of the least squares data fitting problem we take a closer look at the residuals, which we can write as

$$\begin{aligned} r_i = y_i - M(\boldsymbol{x}, t_i) &= \big(y_i - \Gamma(t_i)\big) + \big(\Gamma(t_i) - M(\boldsymbol{x}, t_i)\big) \\ &= e_i + \big(\Gamma(t_i) - M(\boldsymbol{x}, t_i)\big), \\ &\quad i = 1, 2, \ldots, m. \end{aligned} \tag{1.2.5}$$

We see that the ith residual consists of two components: the data error e_i comes from the measurements, while the approximation error $\Gamma(t_i) - M(\boldsymbol{x}, t_i)$ is due to the discrepancy between the pure-data function and the computed fitting model. We emphasize that even if $\Gamma(t)$ and $M(\boldsymbol{x}, t)$ have the same form, there is no guarantee that the estimated parameters $\boldsymbol{x}$ used in $M(\boldsymbol{x}, t)$ will be identical to those underlying the pure-data function $\Gamma(t)$. At any rate, we see from this dichotomy that a good fitting model $M(\boldsymbol{x}, t)$ is one for which the approximation errors are of the same size as the data errors.

Underlying the least squares formulation in (1.2.3) are the assumptions that the data and the errors are independent and that the errors are "white noise." The latter means that all data errors are uncorrelated and of the

same size – or in more precise statistical terms, that the errors e_i have mean zero and identical variance: $\mathcal{E}(e_i) = 0$ and $\mathcal{E}(e_i^2) = \varsigma^2$ for $i = 1, 2, \ldots, m$ (where ς is the standard deviation of the errors).

This ideal situation is not always the case in practice! Hence, we also need to consider the more general case where the standard deviation depends on the index i, i.e.,

$$\mathcal{E}(e_i) = 0, \qquad \mathcal{E}(e_i^2) = \varsigma_i^2, \qquad i = 1, 2, \ldots, m,$$

where ς_i is the standard deviation of e_i. In this case, the maximum likelihood principle in statistics (see Section 1.3), tells us that we should minimize the weighted residuals, with weights equal to the reciprocals of the standard deviations:

$$\boxed{\min_x \sum_{i=1}^m \left(\frac{r_i(\boldsymbol{x})}{\varsigma_i}\right)^2 = \min_x \sum_{i=1}^m \left(\frac{y_i - M(\boldsymbol{x}, t_i)}{\varsigma_i}\right)^2.} \tag{1.2.6}$$

Now consider the expected value of the weighted sum-of-squares:

$$\begin{aligned}
\mathcal{E}\left(\sum_{i=1}^m \left(\frac{r_i(\boldsymbol{x})}{\varsigma_i}\right)^2\right) &= \sum_{i=1}^m \mathcal{E}\left(\frac{r_i(\boldsymbol{x})^2}{\varsigma_i^2}\right) \\
&= \sum_{i=1}^m \mathcal{E}\left(\frac{e_i^2}{\varsigma_i^2}\right) + \sum_{i=1}^m \mathcal{E}\left(\frac{(\Gamma(t_i) - M(\boldsymbol{x}, t_i))^2}{\varsigma_i^2}\right) \\
&= m + \sum_{i=1}^m \frac{\mathcal{E}\big((\Gamma(t_i) - M(\boldsymbol{x}, t_i))^2\big)}{\varsigma_i^2},
\end{aligned}$$

where we have used that $\mathcal{E}(e_i) = 0$ and $\mathcal{E}(e_i^2) = \varsigma_i^2$. The consequence of this relation is the intuitive result that we can allow the expected value of the approximation errors to be larger for those data (t_i, y_i) that have larger standard deviations (i.e., larger errors). Example 4 illustrates the usefulness of this approach. See Chapter 3 in [233] for a thorough discussion on how to estimate weights for a given data set.

We are now ready to state the least squares data fitting problem in terms of matrix-vector notation. We define the matrix $A \in \mathbb{R}^{m \times n}$ and the vectors $\boldsymbol{y}, \boldsymbol{r} \in \mathbb{R}^m$ as follows,

$$A = \begin{pmatrix} f_1(t_1) & f_2(t_1) & \cdots & f_n(t_1) \\ f_1(t_2) & f_2(t_2) & \cdots & f_n(t_2) \\ \vdots & \vdots & & \vdots \\ f_1(t_m) & f_2(t_m) & \cdots & f_n(t_m) \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}, \quad \boldsymbol{r} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix},$$

i.e., $\boldsymbol{y}$ is the vector of observations, $\boldsymbol{r}$ is the vector of residuals and the matrix A is constructed such that the jth column is the jth model basis

function sampled at the abscissas $t_1, t_2, \ldots, t_m$. Then it is easy to see that for the un-weighted data fitting problem we have the relations

$$\boldsymbol{r} = \boldsymbol{y} - A\,\boldsymbol{x} \qquad \text{and} \qquad \rho(\boldsymbol{x}) = \sum_{i=1}^{m} r_i(\boldsymbol{x})^2 = \|\boldsymbol{r}\|_2^2 = \|\boldsymbol{y} - A\,\boldsymbol{x}\|_2^2.$$

Similarly, for the weighted problem we have

$$\rho_W(\boldsymbol{x}) = \sum_{i=1}^{m} \left(\frac{r_i(\boldsymbol{x})}{\varsigma_i} \right)^2 = \|W\,(\boldsymbol{y} - A\,\boldsymbol{x})\|_2^2,$$

with the weighting matrix and weights

$$W = \mathrm{diag}(w_1, \ldots, w_m), \qquad w_i = \varsigma_i^{-1}, \quad i = 1, 2, \ldots, m.$$

In both cases, the computation of the coefficients in the least squares fit is identical to the solution of a linear least squares problem for $\boldsymbol{x}$. Throughout the book we will study these least squares problems in detail and give efficient computational algorithms to solve them.

Example 3. *We return to the NMR data fitting problem from Example 1. For this problem there are 50 measured data points and the model basis functions are*

$$f_1(t) = e^{-\lambda_1 t}, \qquad f_2(t) = e^{-\lambda_2 t}, \qquad f_3(t) = 1,$$

and hence we have $m = 50$ *and* $n = 3$*. In this example the errors in all data points have the same standard deviation* $\varsigma = 0.1$*, so we can use the un-weighted approach. The solution to the* 50×3 *least squares problem is*

$$x_1 = 1.303, \qquad x_2 = 1.973, \qquad x_3 = 0.305.$$

The exact parameters used to generate the data are 1.27*,* 2.04 *and* 0.3*, respectively. These data were then perturbed with random errors. Figure 1.2.1 shows the data together with the least squares fit* $M(\boldsymbol{x}, t)$*; note how the residuals are distributed on both sides of the fit.*

For the data fitting problem in Example 2, we try both the polynomial fit and the trigonometric fit. In the first case the basis functions are the monomials $f_j(t) = t^{n-j}$*, for* $j = 1, \ldots, n = p + 1$*, where* p *is the degree of the polynomial. In the second case the basis functions are the trigonometric functions:*

$$f_1(t) = 1, \quad f_2(t) = \sin(\omega\, t), \quad f_3(t) = \cos(\omega\, t),$$
$$f_4(t) = \sin(2\omega\, t), \quad f_5(t) = \cos(2\omega\, t), \quad \ldots$$

Figure 1.2.2 shows the two fits using a polynomial of degree $p = 8$ *(giving a fit of order* $n = 9$*) and a trigonometric fit with* $n = 9$*. The trigonometric fit looks better. We shall later introduce computational tools that let us investigate this aspect in more rigorous ways.*

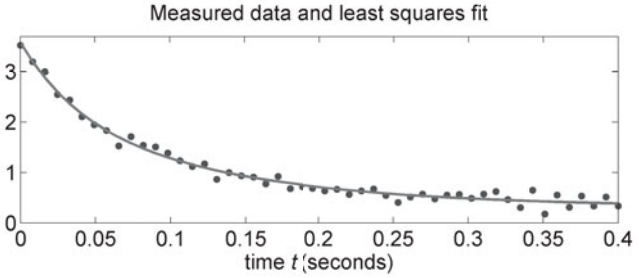

Figure 1.2.1: The least squares fit (solid line) to the measured NMR data (dots) from Figure 1.1.1 in Example 1.

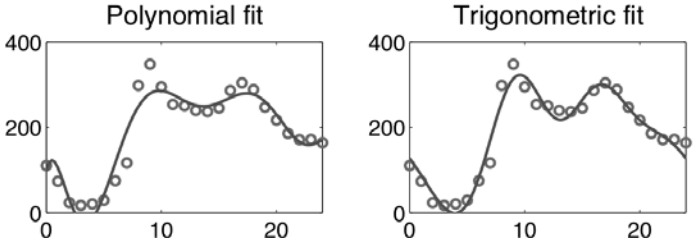

Figure 1.2.2: Two least squares fits (both of order $n = 9$) to the measured NO data from Example 2, using a polynomial (left) and trigonometric functions (right).

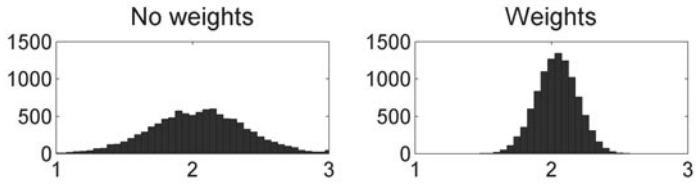

Figure 1.2.3: Histograms of the computed values of x_2 for the modified NMR data in which the first 10 data points have larger errors. The left plot shows results from solving the un-weighted LSQ problem, while the right plot shows the results when weights $w_i = \varsigma_i^{-1}$ are included in the LSQ problem.

Example 4. *This example illustrates the importance of using weights when computing the fit. We use again the NMR data from Example 1, except this time we add larger Gaussian noise to the first 10 data points, with standard deviation* 0.5. *Thus we have* $\varsigma_i = 0.5$ *for* $i = 1, 2, \ldots, 10$ *(the first 10 data with larger errors) and* $\varsigma_i = 0.1$ *for* $i = 11, 12, \ldots, 50$ *(the remaining data with smaller errors). The corresponding weights* $w_i = \varsigma_i^{-1}$ *are therefore* $2, 2, \ldots, 2, 10, 10, \ldots, 10$. *We solve the data fitting problem with and without weights for* $10,000$ *instances of the noise. To evaluate the results, we consider how well we estimate the second parameter* x_2 *whose exact value is* 2.04. *The results are shown in Figure 1.2.3 in the form of histograms of the computed values of* x_2. *Clearly, the weighted fit gives more robust results because it is less influenced by the data with large errors.*

1.3 Maximum likelihood estimation

At first glance the problems of interpolation and data fitting seem to resemble each other. In both cases, we use approximation theory to select a good model (through the model basis functions) for the given data that results in small approximation errors – in the case of data fitting, given by the term $\Gamma(t) - M(\boldsymbol{x}, t)$ for $t \in [t_1, t_m]$. The main difference between the two problems comes from the presence of data errors and the way we deal with these errors.

In data fitting we deliberately avoid interpolating the data and instead settle for less degrees of freedom in the model, in order to reduce the model's sensitivity to errors. Also, it is clear that the data noise plays an important role in data fitting problems, and we should use concepts and tools from statistics to deal with it.

The classical statistical motivation for the least squares fit is based on the maximum likelihood principle. Our presentation follows [13]. We assume that the data are given by (1.2.1), that the errors e_i are unbiased

and uncorrelated, and that each error e_i has a Gaussian distribution with standard deviation ς_i, i.e.,

$$y_i = \Gamma(t_i) + e_i, \qquad e_i \sim \mathcal{N}(0, \varsigma_i^2).$$

Here, $\mathcal{N}(0, \varsigma_i^2)$ denotes the normal (Gaussian) distribution with zero mean and standard deviation σ_i. Gaussian errors arise, e.g., from the measurement process or the measuring devices, and they are also good models of composite errors that arise when several sources contribute to the noise. Then the probability P_i for making the observation y_i is given by

$$P_i = \frac{1}{\varsigma_i\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{y_i - \Gamma(t_i)}{\varsigma_i}\right)^2} = \frac{1}{\varsigma_i\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma_i}\right)^2}.$$

The probability P for making the observed set of measurements $y_1, y_2, \ldots, y_m$ is the product of these probabilities:

$$P = P_1\, P_2\, \cdots\, P_m = \prod_{i=1}^{m} \frac{1}{\varsigma_i\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma_i}\right)^2} = K \prod_{i=1}^{m} e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma_i}\right)^2},$$

where the factor $K = \prod_{i=1}^{m} \left(\varsigma_i\sqrt{2\pi}\right)^{-1}$ is independent of the pure-data function.

Now assume that the pure-data function $\Gamma(t)$ is identical to the fitting model $M(\boldsymbol{x}, t)$, for a specific but unknown parameter vector $\boldsymbol{x}^*$. The probability for the given data $y_1, y_2, \ldots, y_m$ to be produced by the model $M(\boldsymbol{x}, t)$ for an arbitrary parameter vector $\boldsymbol{x}$ is given by

$$P_{\boldsymbol{x}} = K \prod_{i=1}^{m} e^{-\frac{1}{2}\left(\frac{y_i - M(\boldsymbol{x}, t_i)}{\varsigma_i}\right)^2} = K\, e^{-\frac{1}{2}\sum_{i=1}^{m}\left(\frac{y_i - M(\boldsymbol{x}, t_i)}{\varsigma_i}\right)^2}. \tag{1.3.1}$$

The method of maximum likelihood applied to data fitting consists of making the assumption that the given data are more likely to have come from our model $M(\boldsymbol{x}, t)$ – with specific but unknown parameters $\boldsymbol{x}^*$ – than any other model of this form (i.e., with any other parameter vector $\boldsymbol{x}$). The best estimate of the model is therefore the one that maximizes the probability $P_{\boldsymbol{x}}$ given above, as a function of $\boldsymbol{x}$. Clearly, $P_{\boldsymbol{x}}$ is maximized by minimizing the exponent, i.e., the weighted sum-of-squared residuals. We have thus – under the assumption of Gaussian errors – derived the weighted least squares data fitting problem (1.2.6). In practice, we use the same principle when there is a discrepancy between the pure-data function and the fitting model.

The same approach can be used to derive the maximum likelihood fit for other types of errors. As an important example, we consider errors that follow a Laplace distribution, for which the probability of making the

observation y_i is given by

$$P_i = \frac{1}{2\varsigma_i}\, e^{-\frac{|y_i - \Gamma(t_i)|}{\varsigma_i}} = \frac{1}{2\varsigma_i}\, e^{-\frac{|e_i|}{\varsigma_i}},$$

where again ς_i is the standard deviation. The Laplace density function decays slower than the Gaussian density function, and thus the Laplace distribution describes measurement situations where we are more likely to have large errors than in the case of the Gaussian distribution.

Following once again the maximum likelihood approach we arrive at the problem of maximizing the function

$$\begin{aligned} P_{\boldsymbol{x}} &= \widehat{K} \prod_{i=1}^{m} e^{-\frac{|y_i - M(\boldsymbol{x},t_i)|}{\varsigma_i}} \\ &= \widehat{K}\, e^{-\sum_{i=1}^{m}\left|\frac{y_i - M(\boldsymbol{x},t_i)}{\varsigma_i}\right|} \end{aligned}$$

with $\widehat{K} = \prod_{i=1}^{m}\left(2\varsigma_i\right)^{-1}$. Hence, for these errors we should minimize the sum-of-absolute-values of the weighted residuals. This is the linear 1-norm minimization problem that we mentioned in (1.2.4).

While the principle of maximum likelihood is universally applicable, it can lead to complicated or intractable computational problems. As an example, consider the case of Poisson data, where y_i comes from a Poisson distribution, with expected value $\Gamma(t_i)$ and standard deviation $\Gamma(t_i)^{1/2}$. Poisson data typically show up in counting measurements, such as the photon counts underlying optical detectors. Then the probability for making the observation y_i is

$$P_i = \frac{\Gamma(t_i)^{y_i}}{y_i!}\, e^{-\Gamma(t_i)},$$

and hence, we should maximize the probability

$$\begin{aligned} P_{\boldsymbol{x}} &= \prod_{i=1}^{m} \frac{M(\boldsymbol{x},t_i)^{y_i}}{y_i!}\, e^{-M(\boldsymbol{x},t_i)} \\ &= \prod_{i=1}^{m} \frac{1}{y_i!} \prod_{i=1}^{m} M(\boldsymbol{x},t_i)^{y_i}\, e^{-\sum_{i=1}^{m} M(\boldsymbol{x},t_i)}. \end{aligned}$$

Unfortunately, it is computationally demanding to maximize this quantity with respect to $\boldsymbol{x}$, and instead one usually makes the assumption that the Poisson errors for each data y_i are nearly Gaussian, with standard deviation $\sigma_i = \Gamma(t_i)^{1/2} \simeq y_i^{1/2}$ (see, e.g., pp. 342–343 in [146] for a justification of this assumption). Hence, the above weighted least squares approach derived for Gaussian noise, with weights $w_i = y_i^{-1/2}$, will give a good approximation to the maximum likelihood fit for Poisson data.

The Gauss and Laplace errors discussed above are used to model additive errors in the data. We finish this section with a brief look at relative errors, which arise when the size of the error e_i is – perhaps to a good approximation – proportional to the magnitude of the pure data $\Gamma(t_i)$. A straightforward way to model such errors, which fits into the above framework, is to assume that the data y_i can be described by a normal distribution with mean $\Gamma(t_i)$ and standard deviation $\varsigma_i = |\Gamma(t_i)|\,\varsigma$. This "relative Gaussian errors" model can also be written as

$$y_i = \Gamma(t_i)\,(1 + e_i), \qquad e_i \sim \mathcal{N}(0, \varsigma^2). \tag{1.3.2}$$

Then the probability for making the observation y_i is

$$P_i = \frac{1}{|\Gamma(t_i)|\,\varsigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{y_i - \Gamma(t_i)}{\Gamma(t_i)\,\varsigma}\right)^2}.$$

Using the maximum likelihood principle again and substituting the measured data y_i for the unknown pure data $\Gamma(t_i)$, we arrive at the following weighted least squares problem:

$$\min_{\boldsymbol{x}} \sum_{i=1}^{m} \left(\frac{y_i - M(\boldsymbol{x}, t_i)}{y_i}\right)^2 = \min_{\boldsymbol{x}} \|W\,(\boldsymbol{y} - A\,\boldsymbol{x})\|_2^2, \tag{1.3.3}$$

with weights $w_i = y_i^{-1}$.

An alternative formulation, which is suited for problems with positive data $\Gamma(t_i) > 0$ and $y_i > 0$, is to assume that y_i can be described by a log-normal distribution, for which $\log y_i$ has a normal distribution, with mean $\log \Gamma(t_i)$ and standard deviation ς:

$$y_i = \Gamma(t_i)\, e^{\epsilon_i} \quad \Leftrightarrow \quad \log y_i = \log \Gamma(t_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \varsigma^2).$$

In this case we again arrive at a sum-of-squares minimization problem, but now involving the difference of the logarithms of the data y_i and the model $M(\boldsymbol{x}, t_i)$. Even when $M(\boldsymbol{x}, t)$ is a linear model, this is not a linear problem in $\boldsymbol{x}$.

In the above log-normal model with standard deviation ς, the probability P_i for making the observation y_i is given by

$$P_i = \frac{1}{y_i\,\varsigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{\log y_i - \log \Gamma(t_i)}{\varsigma}\right)^2} = \frac{1}{y_i\,\varsigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{\log \hat{y}_i}{\varsigma}\right)^2},$$

with $\hat{y}_i = y_i/\Gamma(t_i)$. Now let us assume that ς is small compared to $\Gamma(t_i)$, such that $y_i \simeq \Gamma(t_i)$ and $\hat{y}_i \simeq 1$. Then, we can write $y_i = \Gamma(t_i)\,(1 + e_i) \Leftrightarrow \hat{y}_i = 1 + e_i$, with $e_i \ll 1$ and $\log \hat{y}_i = e_i + O(e_i^2)$. Hence, the exponential

factor in P_i becomes

$$\begin{aligned} e^{-\frac{1}{2}\left(\frac{\log \hat{y}_i}{\varsigma}\right)^2} &= e^{-\frac{1}{2}\left(\frac{\hat{e}_i + O(e_i^2)}{\varsigma}\right)^2} = e^{-\frac{1}{2}\left(\frac{e_i^2}{\varsigma^2} + \frac{O(e_i^3)}{\varsigma^2}\right)} \\ &= e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma}\right)^2} e^{-\frac{O(e_i^3)}{\varsigma^2}} = e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma}\right)^2} O(1), \end{aligned}$$

while the other factor in P_i becomes

$$\frac{1}{y_i \varsigma \sqrt{2\pi}} = \frac{1}{\Gamma(t_i)\,(1+e_i)\,\varsigma\sqrt{2\pi}} = \frac{1+O(e_i)}{\Gamma(t_i)\,\varsigma\sqrt{2\pi}}.$$

Hence, as long as $\varsigma \ll \Gamma(t_i)$ we have the approximation

$$P_i \simeq \frac{1}{\Gamma(t_i)\,\varsigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma}\right)^2} = \frac{1}{\Gamma(t_i)\,\varsigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{y_i - \Gamma(t_i)}{\Gamma(t_i)\,\varsigma}\right)^2},$$

which is the probability introduced above for the case of "relative Gaussian errors." Hence, for small noise levels $\varsigma \ll |\Gamma(t_i)|$, the two different models for introducing relative errors in the data are practically identical, leading to the same weighted LSQ problem (1.3.3).

1.4 The residuals and their properties

This section focuses on the residuals $r_i = y_i - M(\boldsymbol{x}, t_i)$ for a given fit and how they can be used to analyze the "quality" of the fit $M(\boldsymbol{x}, t)$ that we have computed. Throughout the section we assume that the residuals behave like a time series, i.e., they have a natural ordering $r_1, r_2, \ldots, r_m$ associated with the ordering $t_1 < t_2 < \cdots < t_m$ of the samples of the independent variable t.

As we already saw in Equation (1.2.5), each residual r_i consists of two components – the data error e_i and the approximation error $\Gamma(t_i) - M(\boldsymbol{x}, t_i)$. For a good fitting model, the approximation error should be of the same size as the data errors (or smaller). At the same time, we do not want the residuals to be too small, since then the model $M(\boldsymbol{x}, t)$ may overfit the data: i.e., not only will it capture the behavior of the pure-data function $\Gamma(t)$, but it will also adapt to the errors, which is undesirable.

In order to choose a good fitting model $M(\boldsymbol{x}, t)$ we must be able to analyze the residuals r_i and determine whether the model captures the pure-data function "well enough." We can say that this is achieved when the approximation errors are smaller than the data errors, so that the residuals are practically dominated by the data errors. In that case, some of the statistical properties of the errors will carry over to the residuals. For example, if the noise is white (cf. Section 1.1), then we will expect that the residuals associated with a satisfactory fit show properties similar to white noise.

If, on the other hand, the fitting model does not capture the main behavior of the pure-data function, then we can expect that the residuals are dominated by the approximation errors. When this is the case, the residuals will not have the characteristics of noise, but instead they will tend to behave as a sampled signal, i.e., the residuals will show strong local correlations. We will use the term *trend* to characterize a long-term movement in the residuals when considered as a time series.

Below we will discuss some statistical tests that can be used to determine whether the residuals behave like noise or include trends. These and many other tests are often used in time series analysis and signal processing. Throughout this section we make the following assumptions about the data errors e_i:

- They are random variables with mean zero and identical variance, i.e., $\mathcal{E}(e_i) = 0$ and $\mathcal{E}(e_i^2) = \varsigma^2$ for $i = 1, 2, \ldots, m$.
- They belong to a normal distribution, $e_i \sim \mathcal{N}(0, \varsigma^2)$.

We will describe three tests with three different properties:

- Randomness test: check for randomness of the signs of the residuals.
- Autocorrelation test: check whether the residuals are uncorrelated.
- White noise test: check for randomness of the residuals.

The use of the tools introduced here is illustrated below and in Chapter 11 on applications.

Test for random signs

Perhaps the simplest analysis of the residuals is based on the statistical question: can we consider the signs of the residuals to be random? (Which will often be the case when e_i is white noise with zero mean.) We can answer this question by means of a *run test* from time series analysis; see, e.g., Section 10.4 in [134].

Given a sequence of two symbols – in our case, "+" and "−" for positive and negative residuals r_i – a run is defined as a succession of identical symbols surrounded by different symbols. For example, the sequence "+ + + − − − − + + − − − − − − + + +" has $m = 17$ elements, $n_+ = 8$ pluses, $n_- = 9$ minuses and $u = 5$ runs: + + +, − − −−, ++, − − − − − and +++. The distribution of runs u (not the residuals!) can be approximated by a normal distribution with mean μ_u and standard deviation ς_u given by

$$\mu_u = \frac{2\, n_+\, n_-}{m} + 1, \qquad \varsigma_u^2 = \frac{(\mu_u - 1)\,(\mu_u - 2)}{m - 1}. \tag{1.4.1}$$

With a 5% significance level we will accept the sign sequence as random if

$$z_{\pm} = \frac{|u - \mu_u|}{\varsigma_u} < 1.96 \tag{1.4.2}$$

(other values of the threshold, for other significance levels, can be found in any book on statistics). If the signs of the residuals are not random, then it is likely that trends are present in the residuals. In the above example with 5 runs we have $z_{\pm} = 2.25$, and according to (1.4.2) the sequence of signs is not random.

Test for correlation

Another question we can ask is whether short sequences of residuals are correlated, which is a clear indication of trends. The autocorrelation of the residuals is a statistical tool for analyzing this. We define the *autocorrelation* ϱ of the residuals, as well as the trend threshold T_{ϱ}, as the quantities

$$\varrho = \sum_{i=1}^{m-1} r_i \, r_{i+1}, \qquad T_{\varrho} = \frac{1}{\sqrt{m-1}} \sum_{i=1}^{m} r_i^2. \tag{1.4.3}$$

Since ϱ is the sum of products of neighboring residuals, it is in fact the unit-lag autocorrelation. Autocorrelations with larger lags, or distances in the index, can also be considered. Then, we say that trends are likely to be present in the residuals if the absolute value of the autocorrelation exceeds the trend threshold, i.e., if $|\varrho| > T_{\varrho}$. Similar techniques, based on shorter sequences of residuals, are used for placing knots in connection with spline fitting; see Chapter 6 in [125].

We note that in some presentations, the mean of the residuals is subtracted before computing ϱ and T_{ϱ}. In our applications this should not be necessary, as we assume that the errors have zero mean.

Test for white noise

Yet another question we can ask is whether the sequence of residuals behaves like white noise, which can be answered by means of the normalized cumulative periodogram. The underlying idea is that white noise has a flat spectrum, i.e., all frequency components in the discrete Fourier spectrum have the same probability; hence, we must determine whether this is the case. Let the complex numbers $\hat{r}_k$ denote the components of the discrete Fourier transform of the residuals, i.e.,

$$\hat{r}_k = \sum_{i=1}^{m} r_i \, e^{(2\pi \, \hat{\imath} \, (i-1)(k-1)/m)}, \qquad k = 1, \ldots, m,$$

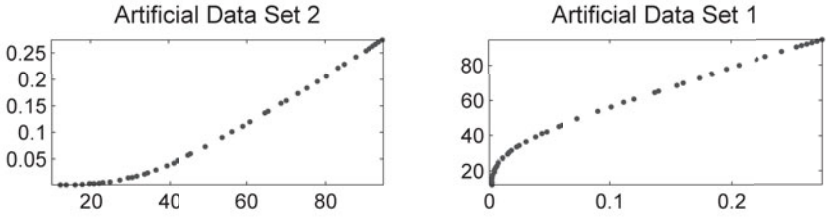

Figure 1.4.1: Two artificially created data sets used in Example 5. The second data set is inspired by the data on p. 60 in [125].

where $\hat{\imath}$ denotes the imaginary unit. Our indices are in the range $1, \ldots, m$ and thus shifted by 1 relative to the range $0, \ldots, m-1$ that is common in signal processing. Note that $\hat{r}_1$ is the sum of the residuals (called the DC component in signal processing), while $\hat{r}_{q+1}$ with $q = \lfloor m/2 \rfloor$ is the component of the highest frequency. The squared absolute values $|\hat{r}_1|^2, |\hat{r}_2|^2, \ldots, |\hat{r}_{q+1}|^2$ are known as the periodogram (in statistics) or the power spectrum (in signal processing). Then the normalized cumulative periodogram consists of the q numbers

$$c_i = \frac{|\hat{r}_2|^2 + |\hat{r}_3|^2 + \cdots + |\hat{r}_{i+1}|^2}{|\hat{r}_2|^2 + |\hat{r}_3|^2 + \cdots + |\hat{r}_{q+1}|^2}, \qquad i = 1, \ldots, q, \qquad q = \lfloor m/2 \rfloor,$$

which form an increasing sequence from 0 to 1. Note that the sums exclude the first term in the periodogram.

If the residuals are white noise, then the expected values of the normalized cumulative periodogram lie on a straight line from $(0, 0)$ to $(q, 1)$. Any realization of white noise residuals should produce a normalized cumulative periodogram close to a straight line. For example, with the common 5% significance level from statistics, the numbers c_i should lie within the Kolmogorov-Smirnoff limit $\pm 1.35/q$ of the straight line. If the maximum deviation $\max_i\{|c_i - i/q|\}$ is smaller than this limit, then we recognize the residual as white noise.

Example 5. ***Residual analysis.*** *We finish this section with an example that illustrates the above analysis techniques. We use the two different data sets shown in Figure 1.4.1; both sets are artificially generated (in fact, the second set is the first set with the t_i and y_i values interchanged). In both examples we have $m = 43$ data points, and in the test for white noise we have $q = 21$ and $1.35/q = 0.0643$. The fitting model $M(\boldsymbol{x}, t)$ is the polynomial of degree $p = n - 1$.*

For fitting orders $n = 2, 3, \ldots, 9$, Figure 1.4.2 shows the residuals and the normalized cumulative periodograms, together with $z_\pm$ (1.4.2), the ratios

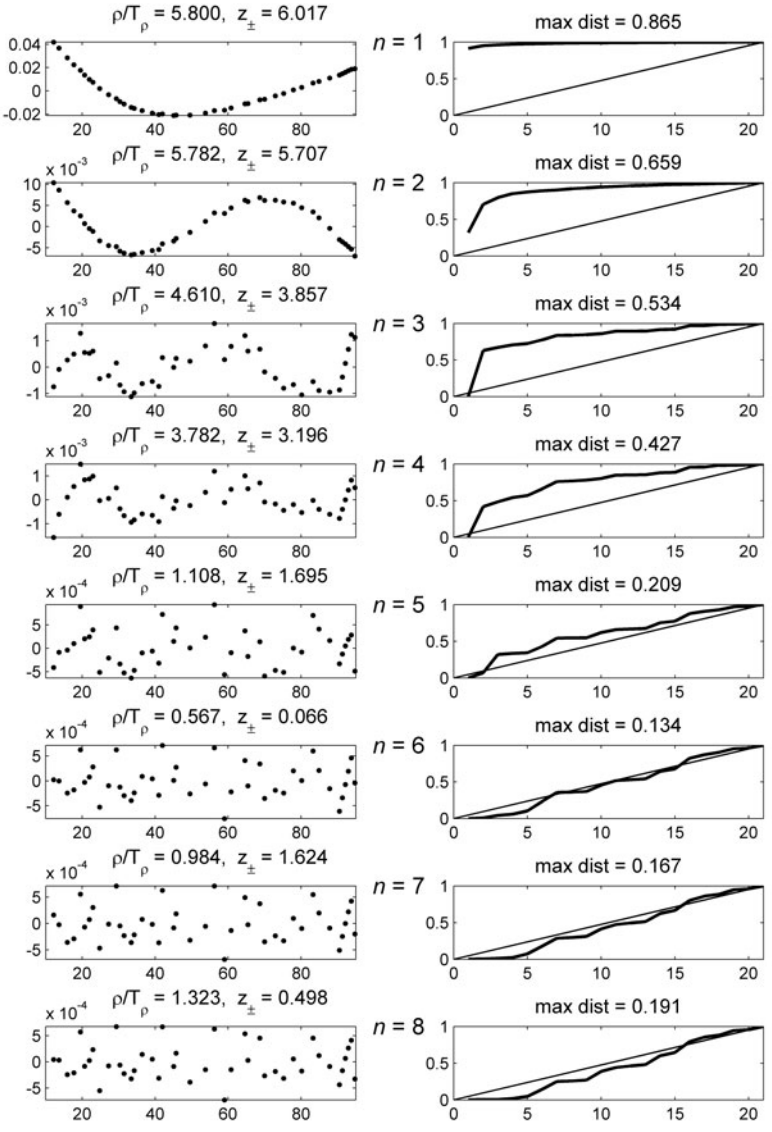

Figure 1.4.2: Residual analysis for polynomial fits to artificial data set 1.

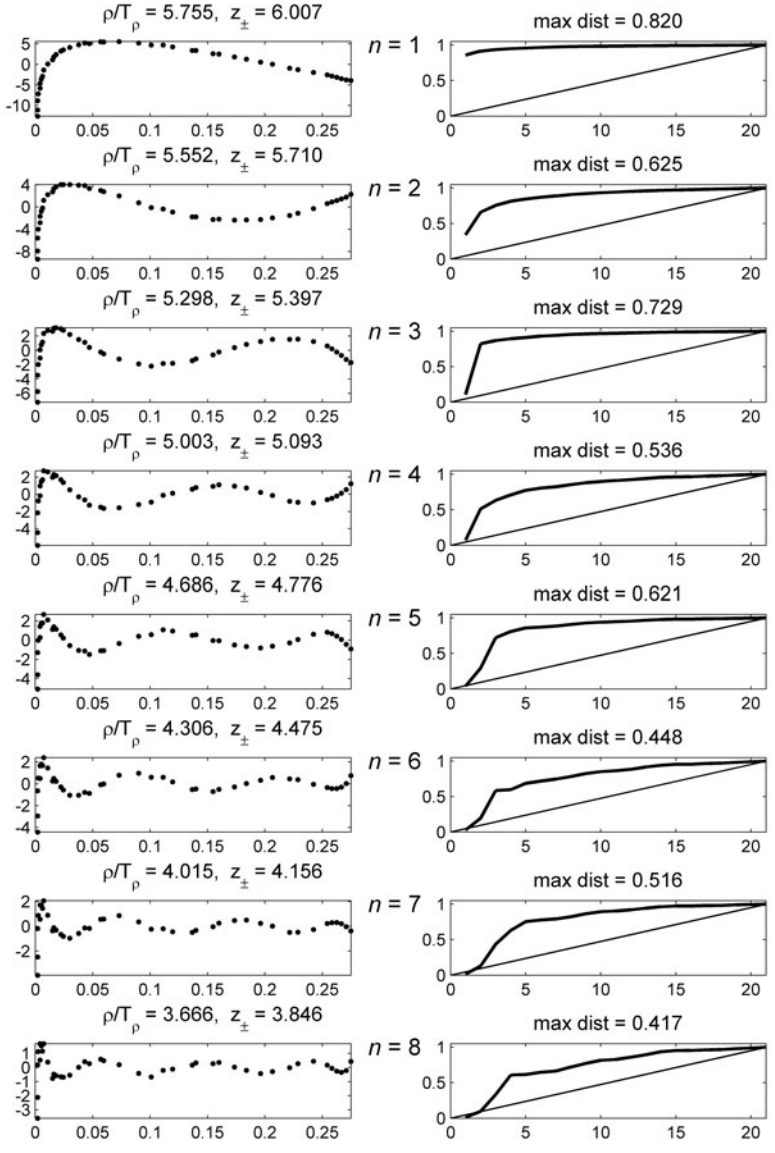

Figure 1.4.3: Residual analysis for polynomial fits to artificial data set 2.

ϱ/T_ϱ from (1.4.3) and the maximum distance of the normalized cumulative periodogram to the straight line. A visual inspection of the residuals in the left part of the figure indicates that for small values of n the polynomial model does not capture all the information in the data, as there are obvious trends in the residuals, while for $n \geq 5$ the residuals appear to be more random. The test for random signs confirms this: for $n \geq 5$ the numbers $z_\pm$ are less than 1.96, indicating that the signs of the residuals could be considered random. The autocorrelation analysis leads to approximately the same conclusion: for $n = 6$ and 7 the absolute value of the autocorrelation ϱ is smaller than the threshold T_ϱ.

The normalized cumulative periodograms are shown in the right part of Figure 1.4.2. For small values, the curves rise fast toward a flat part, showing that the residuals are dominated by low-frequency components. The closest we get to a straight line is for $n = 6$, but the maximum distance 0.134 to the straight line is still too large to clearly signify that the residuals are white noise. The conclusion from these three tests is nevertheless that $n = 6$ is a good choice of the order of the fit.

Figure 1.4.3 presents the residual analysis for the second data set. A visual inspection of the residuals clearly shows that the polynomial model is not well suited for this data set – the residuals have a slowly varying trend for all values of n. This is confirmed by the normalized cumulative periodograms that show that the residuals are dominated by low-frequency components. The random-sign test and the autocorrelation analysis also give a clear indication of trends in the residuals.

1.5 Robust regression

The least squares fit introduced in this chapter is convenient and useful in a large number of practical applications – but it is not always the right choice for a data fitting problem. In fact, we have already seen in Section 1.3 that the least squares fit is closely connected to the assumption about Gaussian errors in the data. There we also saw that other types of noise, in the framework of maximum likelihood estimation, lead to other criteria for a best fit – such as the sum-of-absolute-values of the residuals (the 1-norm) associated with the Laplace distribution for the noise. The more dominating the "tails" of the probablility density function for the noise, the more important it is to use another criterion than the least squares fit.

Another situation where the least squares fit is not appropriate is when the data contain *outliers*, i.e., observations with exceptionally large errors and residuals. We can say that an outlier is a data point (t_i, y_i) whose value y_i is unusual compared to its predicted value (based on all the reliable data points). Such outliers may come from different sources:

- The data errors may come from more than one statistical distribution. This could arise, e.g., in an astronomical CCD camera, where we have Poisson noise (or "photon noise") from the incoming light, Gaussian noise from the electronic circuits (amplifier and A/D-converter), and occasional large errors from cosmic radiation (so-called cosmic ray events).
- The outliers may be due to data recording errors arising, e.g., when the measurement device has a malfunction or the person recording the data makes a blunder and enters a wrong number.

A manual inspection can sometimes be used to delete blunders from the data set, but it may not always be obvious which data are blunders or outliers. Therefore we prefer to have a mathematical formulation of the data fitting problem that handles outliers in such a way that all data are used, and yet the outliers do not have a deteriorating influence on the fit. This is the goal of *robust regression.* Quoting from [82] we say that "an estimator or statistical procedure is robust if it provides useful information even if some of the assumptions used to justify the estimation method are not applicable."

Example 6. ***Mean and median.*** *Assume we are given $n-1$ samples $z_1, \ldots, z_{n-1}$ from the same distribution and a single sample z_n that is an outlier. Clearly, the arithmetic mean $\frac{1}{n}(z_1 + z_2 + \cdots + z_n)$ is not a good estimate of the expected value because the outlier constributes with the same weight as all the other data points. On the other hand, the median gives a robust estimate of the expected value since it is insensitive to a few outliers; we recall that if the data are sorted then the median is $z_{(n+1)/2}$ if n is odd, and $\frac{1}{2}(z_{n/2} + z_{n/2+1})$ if n is even.*

The most common method for robust data fitting – or robust regression, as statisticians call it – is based on the principle of M-estimation introduced by Huber [130], which can be considered as a generalization of maximum likelihood estimation. Here we consider un-weighted problems only (the extension to weighted problems is straightforward). The underlying idea is to replace the sum of squared residuals in (1.2.3) with the sum of some function of the residuals:

$$\text{Robust fit:} \quad \min_{\boldsymbol{x}} \sum_{i=1}^{m} \rho\big(r_i(\boldsymbol{x})\big) = \min_{\boldsymbol{x}} \sum_{i=1}^{m} \rho\big(y_i - M(\boldsymbol{x}, t_i)\big), \tag{1.5.1}$$

where the function ρ defines the contribution of each residual to the function to be minimized. In particular, we obtain the least squares fit when $\rho(r) = \frac{1}{2}r^2$. The function ρ must satisfy the following criteria:

1. Non-negativity: $\rho(r) \geq 0 \ \forall r$.

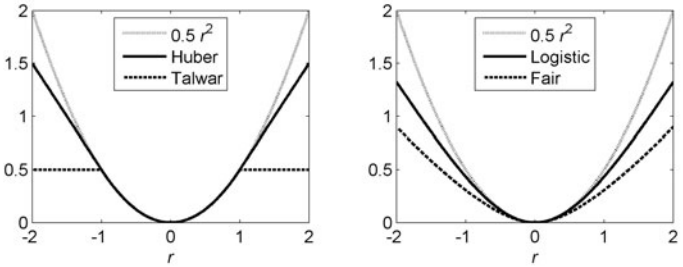

Figure 1.5.1: Four functions $\rho(r)$ used in the robust data fitting problem. All of them increase slower than $\frac{1}{2}r^2$ that defines the LSQ problem and thus they lead to robust data fitting problems that are less sensitive to outliers.

2. Zero only when the argument is zero: $\rho(r) = 0 \Leftrightarrow r = 0$.
3. Symmetry: $\rho(-r) = \rho(r)$.
4. Monotonicity: $\rho(r') \geq \rho(r)$ for $r' \geq r$.

Some well-known examples of the function ρ are (cf. [82, 171]):

$$\text{Huber}: \quad \rho(r) = \begin{cases} \frac{1}{2}r^2, & |r| \leq \beta \\ \beta|r| - \frac{1}{2}\beta^2, & |r| > \beta \end{cases} \tag{1.5.2}$$

$$\text{Talwar}: \quad \rho(r) = \begin{cases} \frac{1}{2}r^2, & |r| \leq \beta \\ \frac{1}{2}\beta^2, & |r| > \beta \end{cases} \tag{1.5.3}$$

$$\text{Bisquare}: \quad \rho(r) = \beta^2 \log\left(\cosh\left(\frac{z}{\beta}\right)\right) \tag{1.5.4}$$

$$\text{Logistic}: \quad \rho(r) = \beta^2 \left(\frac{|r|}{\beta} - \log\left(1 + \frac{|r|}{\beta}\right)\right) \tag{1.5.5}$$

Note that all four functions include a problem-dependent positive parameter β that is used to control the behavior of the function for "large" values of r, corresponding to the outliers. Figure 1.5.1 shows these functions for the case $\beta = 1$, and we see that all of them increase slower than the function $\frac{1}{2}r^2$, which underlies the LSQ problem. This is precisely why they lead to a robust data fitting problem whose solution is less sensitive to outliers than the LSQ solution. The parameter β should be chosen from our knowledge of the standard deviation ς of the noise; if ς is not known, then it can be estimated from the fit as we will discuss in Section 2.2.

It appears that the choice of function ρ for a given problem relies mainly on experience with the specific data for that problem. Still, the Huber

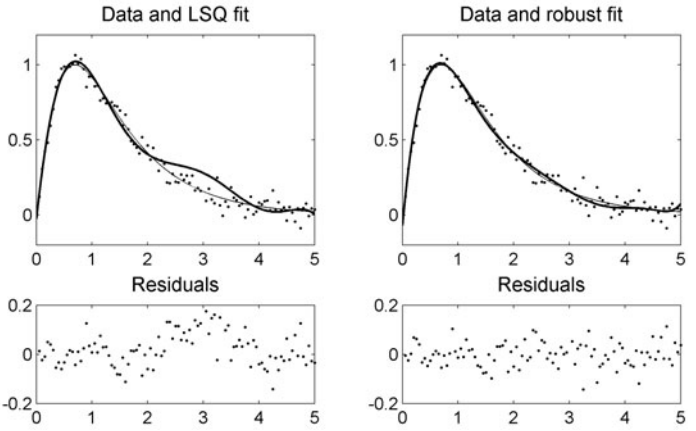

Figure 1.5.2: The pure-data function $\Gamma(t)$ (thin line) and the data with Gaussian noise (dots); the outlier $(t_{60}, y_{60}) = (3, 2.5)$ is outside the plot. The fitting model $M(\boldsymbol{x}, t)$ is a polynomial with $n = 9$. Left: the LSQ fit and the corresponding residuals; this fit is dramatically influenced by the outlier. Right: the robust Huber fit, using $\beta = 0.025$, together with the residuals; this is a much better fit to the given data because it approximates the pure-data function well.

function has attained widespread use, perhaps due to the natural way it distinguishes between large and small residuals:

- Small residual satisfying $|r_i| \leq \beta$ are treated in the same way as in the LSQ fitting problem; if there are no outliers then we obtain the LSQ solution.

- Large residuals satisfying $|r_i| > \beta$ are essentially treated as $|r_i|$ and the robust fit is therefore not so sensitive to the corresponding data points.

Thus, robust regression is a compromise between excluding the outliers entirely from the analysis and including all the data points and treating all of them equally in the LSQ regression. The idea of robust regression is to weight the observations differently based on how well behaved these observations are. For an early use in seismic data processing see [48, 219].

Example 7. ***Robust data fitting with the Huber function.*** *This example illustrates that the Huber function gives a more robust fit than the LSQ fit. The pure-data funtion* $\Gamma(t)$ *is given by*

$$\Gamma(t) = \sin\left(\pi e^{-t}\right), \qquad 0 \leq t \leq 5.$$

We use $m = 100$ *data points with* $t_i = 0.05i$*, and we add Gaussian noise with standard deviation* $\varsigma = 0.05$*. Then we change the 60th data point to an outlier with* $(t_{60}, y_{60}) = (3, 2.5)$*; Figure 1.5.2 shows the function* $\Gamma(t)$ *and the noisy data. We note that the outlier is located outside the plot.*

As fitting model $M(\boldsymbol{x}, t)$ *we use a polynomial with* $n = 9$ *and the left part of Figure 1.5.2 shows the least squares fit with this model, together with the corresponding residuals. Clearly, this fit is dramatically influenced by the outlier, which is evident from the plot of the fit and as well by the behavior of the residuals, which exhihit a strong positive "trend" in the range* $2 \leq t \leq 4$*. This illustrates the inability of the LSQ fit to handle outliers in a satisfactory way.*

The right part of Figure 1.5.2 shows the robust Huber fit, with parameter $\beta = 0.025$ *(this parameter is chosen to reflect the noise level in the data). The resulting fit is not influenced by the outlier, and the residuals do not seem to exhibit any strong "trend." This is a good illustration of robust regression.*

In Section 9.5 we describe numerical algorithms for computing the solutions to robust data fitting problems.

This page intentionally left blank

Chapter 2

The Linear Least Squares Problem

This chapter covers some of the basic mathematical facts of the linear least squares problem (LSQ), as well as some important additional statistical results for data fitting. We introduce the two formulations of the least squares problem: the linear system of normal equations and the optimization problem form.

The computation of the LSQ solution via an optimization problem has two aspects: simplification of the problem structure and actual minimization. In this and in the next chapter we present a number of matrix factorizations, for both full-rank and rank-deficient problems, which transform the original problem to one easier to solve. The QR factorization is emphasized, for both the analysis and the solution of the LSQ problem, while in the last section we look into the more expensive complete factorizations.

Some very interesting historical paper on Gaussian elimination that includes also least squares problems can be found in Grcar [109, 110].

2.1 Linear least squares problem formulation

As we saw in the previous chapter, underlying the linear (and possibly weighted) least squares data fitting problem is the linear least squares problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2 \qquad \text{or} \qquad \min_{\boldsymbol{x}} \|W(\boldsymbol{b} - A\,\boldsymbol{x})\|_2,$$

where $A \in \mathbb{R}^{m\times n}$ is the matrix with samples of the model basis functions, x is the vector of parameters to be determined, the right-hand side $\boldsymbol{b}$ is the vector of observations and W is a diagonal weight matrix (possibly the identity matrix). Since the weights can always be absorbed into A and b

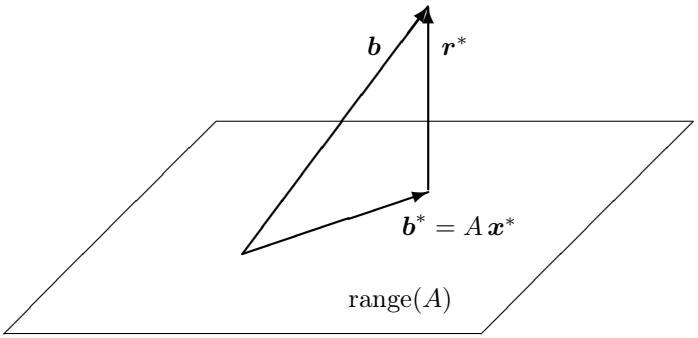

Figure 2.1.1: The geometric interpretation of the linear least squares solution $\boldsymbol{x}^*$. The plane represents the range of A, and if the vector $\boldsymbol{b}$ has a component outside this subspace, then we have an inconsistent system. Moreover, $\boldsymbol{b}^* = A\,\boldsymbol{x}^*$ is the orthogonal projection of $\boldsymbol{b}$ on range(A), and $\boldsymbol{r}^*$ is the LSQ residual vector.

in the mathematical formulation we can, without loss of generality, restrict our discussion to the un-weighted case. Also, from this point on, when discussing the generic linear least squares problem, we will use the notation $\boldsymbol{b}$ for the right-hand side (instead of $\boldsymbol{y}$ that we used in Chapter 1), which is more common in the LSQ literature.

Although most of the material in this chapter is also applicable to the underdetermined case ($m < n$), for notational simplicity we will always consider the overdetermined case $m \geq n$. We denote by r the rank of A, and we consider both full-rank and rank-deficient problems. Thus we always have $r \leq n \leq m$ in this chapter.

It is appropriate to remember at this point that an $m \times n$ matrix A is always a representation of a linear transformation $\boldsymbol{x} \rightarrow A\,\boldsymbol{x}$ with $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$, and therefore there are two important subspaces associated with it: The *range* or *column space*,

$$\text{range}(A) = \{\boldsymbol{z} \in \mathbb{R}^m \mid \boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{z} = A\,\boldsymbol{x}\}\,,$$

and its orthogonal complement, the *null space* of A^T:

$$\text{null}(A^T) = \left\{\boldsymbol{y} \in \mathbb{R}^m \mid A^T\boldsymbol{y} = \boldsymbol{0}\right\}.$$

When A is square and has full rank, then the LSQ problem $\min_{\boldsymbol{x}} \|A\,\boldsymbol{x} - \boldsymbol{b}\|_2$ reduces to the linear system of equations $A\,\boldsymbol{x} = \boldsymbol{b}$. In all other cases, due to the data errors, it is highly probable that the problem is *inconsistent*, i.e., $\boldsymbol{b} \notin \text{range}(A)$, and as a consequence there is no exact solution, i.e., no coefficients x_j exist that express $\boldsymbol{b}$ as a linear combination of columns of A.

Instead, we can find the coefficients x_j for a vector $\boldsymbol{b}^*$ in the range of A and "closest" to $\boldsymbol{b}$. As we have seen, for data fitting problems it is natural to use the Euclidean norm as our measure of closeness, resulting in the least squares problem

$$\boxed{\text{Problem LSQ}: \quad \min_x \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2, \quad \mathrm{A} \in \mathbb{R}^{m\times n}, \quad r \le n \le m} \tag{2.1.1}$$

with the corresponding residual vector $\boldsymbol{r}$ given by

$$\boldsymbol{r} = \boldsymbol{b} - A\,\boldsymbol{x}. \tag{2.1.2}$$

See Figure 2.1.1 for a geometric interpretation. The minimizer, i.e., the least squares solution – which may not be unique, as it will be seen later – is denoted by $\boldsymbol{x}^*$. We note that the vector $\boldsymbol{b}^* \in \text{range}(A)$ mentioned above is given by $\boldsymbol{b}^* = A\,\boldsymbol{x}^*$.

The LSQ problem can also be looked at from the following point of view. When our data are contaminated by errors, then the data are not in the span of the model basis functions $f_j(t)$ underlying the data fitting problem (cf. Chapter 1). In that case the data vector $\boldsymbol{b}$ cannot – and should not – be precisely "predicted" by the model, i.e., the columns of A. Hence, it must be perturbed by a minimum amount $\boldsymbol{r}$, so that it can then be "represented" by A, in the form of $\boldsymbol{b}^* = A\,\boldsymbol{x}^*$. This approach will establish a viewpoint used in Section 7.3 to introduce the total least squares problem.

As already mentioned, there are good statistical reasons to use the Euclidean norm. The underlying statistical assumption that motivates this norm is that the vector $\boldsymbol{r}$ has random error elements, uncorrelated, with zero mean and a common variance. This is justified by the following theorem.

Theorem 8. *(Gauss-Markov) Consider the problem of fitting a model $M(\boldsymbol{x}, t)$ with the n-parameter vector $\boldsymbol{x}$ to a set of data $b_i = \Gamma(t_i) + e_i$ for $i = 1, \ldots, m$ (see Chapter 1 for details).*

In the case of a linear model $\boldsymbol{b} = A\,\boldsymbol{x}$, if the errors are uncorrelated with mean zero and constant variance ς^2 (not necessarily normally distributed) and assuming that the $m \times n$ matrix A obtained by evaluating the model at the data abscissas $\{t_i\}_{i=1,\ldots,m}$ has full rank n, then the best linear unbiased estimator is the least squares estimator $\boldsymbol{x}^*$, obtained by solving the problem $\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2$.

For more details see [20] Theorem 1.1.1. Recall also the discussion on maximum likelihood estimation in Chapter 1. Similarly, for nonlinear models, if the errors e_i for $i = 1, \ldots, m$ have a normal distribution, the unknown parameter vector $\boldsymbol{x}$ estimated from the data using a least squares criterion is the maximum likelihood estimator.

There are also clear mathematical and computational advantages associated with the Euclidean norm: the objective function in (2.1.1) is differentiable, and the resulting gradient system of equations has convenient properties. Since the Euclidean norm is preserved under orthogonal transformations, this gives rise to a range of stable numerical algorithms for the LSQ problem.

Theorem 9. *A necessary and sufficient condition for $\boldsymbol{x}^*$ to be a minimizer of $\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2$ is that it satisfies*

$$A^T(\boldsymbol{b} - A\,\boldsymbol{x}) = \boldsymbol{0}. \tag{2.1.3}$$

Proof. The minimizer of $\rho(\boldsymbol{x}) = \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2$ must satisfy $\nabla\rho(\boldsymbol{x}) = \boldsymbol{0}$, i.e., $\partial\rho(\boldsymbol{x})/\partial x_k = 0$ for $k = 1, \ldots, n$. The kth partial derivative has the form

$$\begin{aligned} \frac{\partial\rho(\boldsymbol{x})}{\partial x_k} &= \sum_{i=1}^{m} 2\Big(b_i - \sum_{j=1}^{n} x_j a_{ij}\Big)(-a_{ik}) = -2\sum_{i=1}^{m} r_i\, a_{ik} \\ &= -2\,\boldsymbol{r}^T A(:,k) = -2\,A(:,k)^T\boldsymbol{r}, \end{aligned}$$

where $A(:,k)$ denotes the kth column of A. Hence, the gradient can be written as

$$\nabla\rho(\boldsymbol{x}) = -2\,A^T\boldsymbol{r} = -2\,A^T(\boldsymbol{b} - A\,\boldsymbol{x})$$

and the requirement that $\nabla\rho(\boldsymbol{x}) = \boldsymbol{0}$ immediately leads to (2.1.3). □

Definition 10. *The two conditions* (2.1.2) *and* (2.1.3) *can be written as a symmetric $(m+n)\times(m+n)$ system in $\boldsymbol{x}$ and $\boldsymbol{r}$, the so-called augmented system:*

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix}. \tag{2.1.4}$$

This formulation preserves any special structure that A might have, such as sparsity. Also, it is the formulation used in an iterative refinement procedure for the LSQ solution (discussed in Section 4.5), because of the relevance it gives to the residual.

Theorem 9 leads to the *normal equations* for the solution x^* of the least squares problem:

$$\boxed{\text{Normal equations: } A^T A\,\boldsymbol{x} = A^T\boldsymbol{b}.} \tag{2.1.5}$$

The normal equation matrix A^TA, which is sometimes called the Grammian, is square, symmetric and additionally:

- If $r = n$ (A has full rank), then A^TA is positive definite and the LSQ problem has a unique solution. (Since the Hessian for the least squares problem is equal to $2A^TA$, this establishes the uniqueness of $\boldsymbol{x}^*$.)

- If $r < n$ (A is rank deficient), then $A^T A$ is non-negative definite. In this case, the set of solutions forms a linear manifold of dimension $n - r$ that is a translation of the subspace null(A).

Theorem 9 also states that the residual vector of the LSQ solution lies in null(A^T). Hence, the right-hand side $\boldsymbol{b}$ can be decomposed into two orthogonal components

$$\boldsymbol{b} = A\,\boldsymbol{x} + \boldsymbol{r},$$

with $A\,\boldsymbol{x} \in$ range(A) and $\boldsymbol{r} \in$ null(A^T), i.e., $A\,\boldsymbol{x}$ is the orthogonal projection of $\boldsymbol{b}$ onto range(A) (the subspace spanned by the columns of A) and $\boldsymbol{r}$ is orthogonal to range(A).

Example 11. *The normal equations for the NMR problem in Example 1 take the form*

$$\begin{pmatrix} 2.805 & 4.024 & 5.055 \\ 4.024 & 8.156 & 1.521 \\ 5.055 & 1.521 & 50 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 13.14 \\ 25.98 \\ 51.87 \end{pmatrix},$$

giving the least squares solution $\boldsymbol{x}^* = (\,1.303\,,\,1.973\,,\,0.305\,)^T$.

Example 12. ***Simplified NMR problem.*** *In the NMR problem, let us assume that we know that the constant background is 0.3, corresponding to fixing* $x_3 = 0.3$. *The resulting* 2×2 *normal equations for* x_1 *and* x_2 *take the form*

$$\begin{pmatrix} 2.805 & 4.024 \\ 4.024 & 8.156 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 10.74 \\ 20.35 \end{pmatrix}$$

and the LSQ solution to this simplified problem is $x_1 = 1.287$ *and* $x_2 = 1.991$. *Figure 2.1.2 illustrates the geometry of the minimization associated with the simplified LSQ problem for the two unknowns* x_1 *and* x_2. *The left plot shows the residual norm surface as a function of* x_1 *and* x_2, *and the right plot shows the elliptic contour curves for this surface; the unique minimum – the LSQ solution – is marked with a dot.*

In the rank-deficient case the LSQ solution is not unique, but one can reduce the solution set by imposing additional constraints. For example, the linear least squares problem often arises from a linearization of a nonlinear least squares problem, and it may be of interest then to impose the additional constraint that the solution has minimal l_2-norm, $\widehat{\boldsymbol{x}} = \min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2$, so that the solution stays in the region where the linearization is valid. Because the set of all minimizers is convex there is a unique solution. Another reason for imposing minimal length is stability, as we will see in the section on regularization.

For data approximation problems, where we are free to choose the model basis functions $f_j(t)$, cf. (1.2.2), one should do it in a way that

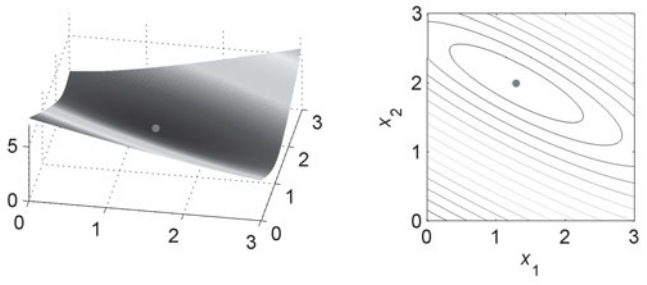

Figure 2.1.2: Illustration of the LSQ problem for the simplified NMR problem. Left: the residual norm as a function of the two unknowns x_1 and x_2. Right: the corresponding contour lines for the residual norm.

gives A full rank. A necessary condition is that the (*continuous*) functions $f_1(t), \ldots, f_n(t)$ are linearly independent, but furthermore, they have to define linearly independent vectors when evaluated on the *specific discrete set of abscissas*. More formally:

A necessary and sufficient condition for the matrix A to have full rank is that the model basis functions be linearly independent over the abscissas $t_1, \ldots, t_m$:

$$\sum_{j=1}^{n} \alpha_j f_j(t_i) = 0 \text{ for } i = 1, \ldots, m \quad \Leftrightarrow \quad \alpha_j = 0 \text{ for } j = 1, ..., n.$$

Example 13. *Consider the linearly independent functions* $f_1(t) = \sin(t)$, $f_2(t) = \sin(2t)$ *and* $f_3(t) = \sin(3t)$*; if we choose the data abscissas* $t_i = \pi/4 + i\pi/2$, $i = 1, \ldots, m$, *the matrix* A *has rank* $r = 2$, *whereas the same functions generate a full-rank matrix* A *when evaluated on the abscissas* $t_i = \pi(i/m)$, $i = 1 \ldots, m - 1$.

An even stronger requirement is that the model basis functions $f_j(t)$ be such that the columns of A are orthogonal.

Example 14. *In general, for data fitting problems, where the model basis functions* $f_j(t)$ *arise from the underlying model, the properties of the matrix* A *are dictated by these functions. In the case of polynomial data fitting, it is possible to choose the functions* $f_j(t)$ *so that the columns of* A *are orthogonal, as described by Forsythe [81], which simplifies the computation of the LSQ solution. The key is to choose a clever representation of the fitting polynomials, different from the standard one with the monomials:*

$f_j(t) = t^{j-1},\ j = 1, ..., n$, such that the sampled polynomials satisfy

$$\sum_{i=1}^{m} f_j(t_i)\, f_k(t_i) = 0 \quad \text{for } j \neq k. \tag{2.1.6}$$

When this is the case we say that the functions are orthogonal over the given abscissas. This is satisfied by the family of orthogonal polynomials defined by the recursion:

$$\begin{aligned} f_1(t) &= 1 \\ f_2(t) &= t - \alpha_1 \\ f_{j+1}(t) &= (t - \alpha_j)\, f_j(t) - \beta_j f_{j-1}(t), \qquad j = 2, \ldots n-1, \end{aligned}$$

where the constants are given by

$$\begin{aligned} \alpha_j &= \frac{1}{s_j^2} \sum_{i=1}^{m} t_i\, f_j(t_i)^2, \qquad j = 0, 1, \ldots, n-1 \\ \beta_j &= \frac{s_j^2}{s_{j-1}^2}, \qquad j = 0, 1, \ldots, n-1 \\ s_j^2 &= \sum_{i=1}^{m} f_j(t_i)^2, \qquad j = 2, \ldots n, \end{aligned}$$

i.e., s_j is the l_2-norm of the jth column of A. These polynomials satisfy (2.1.6), hence, the normal equations matrix $A^T A$ is diagonal, and it follows that the LSQ coefficients are given by

$$x_j^* = \frac{1}{s_j^2} \sum_{i=1}^{m} y_i\, f_j(t_i), \qquad j = 1, \ldots, n.$$

When A has full rank, it follows from the normal equations (2.1.5) that we can write the least squares solution as

$$\boldsymbol{x}^* = (A^T A)^{-1} A^T \boldsymbol{b},$$

which allows us to analyze the solution and the residual vector in statistical terms. Consider the case where the data errors e_i are independent, uncorrelated and have identical standard deviations ς, meaning that the covariance for $\boldsymbol{b}$ is given by

$$\mathrm{Cov}(\boldsymbol{b}) = \varsigma^2\, I_m,$$

since the errors e_i are independent of the exact $\Gamma(t_i)$. Then a standard result in statistics says that the covariance matrix for the LSQ solution is

$$\mathrm{Cov}(\boldsymbol{x}^*) = (A^T A)^{-1} A^T \mathrm{Cov}(\boldsymbol{b})\, A\, (A^T A)^{-1} = \varsigma^2\, (A^T A)^{-1}.$$

We see that the unknown coefficients in the fit – the elements of $\boldsymbol{x}^*$ – are uncorrelated if and only if A^TA is a diagonal matrix, i.e., when the columns of A are orthogonal. This is the case when the model basis functions are orthogonal over the abscissas $t_1, \ldots, t_m$; cf. (2.1.6).

Example 15. ***More data gives better accuracy.*** *Intuitively we expect that if we increase the number of data points then we can compute a more accurate LSQ solution, and the present example confirms this. Specifically we give an asymptotic analysis of how the solution's variance depends on the number m of data points, in the case of linear data fitting. There is no assumption about the distribution of the abscissas t_i except that they belong to the interval $[a, b]$ and appear in increasing order. Now let $h_i = t_i - t_{i-1}$ for $i = 2, \ldots, m$ and let $h = (b-a)/m$ denote the average spacing between the abscissas. Then for $j, k = 1, \ldots, n$ the elements of the normal equation matrix can be approximated as*

$$\begin{aligned}(A^TA)_{jk} &= \sum_{i=1}^{m} h_i^{-1} f_j(t_i) f_k(t_i) h_i \simeq \frac{1}{h} \sum_{i=1}^{m} f_j(t_i) f_k(t_i) h_i \\ &\simeq \frac{m}{b-a} \int_a^b f_j(t) f_k(t)\, dt,\end{aligned}$$

and the accuracy of these approximations increases as m increases. Hence, if F denotes the matrix whose elements are the scaled inner products of the model basis functions,

$$F_{jk} = \frac{1}{b-a} \int_a^b f_j(t) f_k(t) dt, \qquad i, j = 1, \ldots, n,$$

then for large m the normal equation matrix approximately satisfies

$$A^TA \simeq m\, F \qquad \Leftrightarrow \qquad (A^TA)^{-1} \simeq \frac{1}{m}\, F^{-1},$$

where the matrix F is independent of m. Hence, the asymptotic result (as m increases) is that no matter the choice of abscissas and basis functions, as long as A^TA is invertible we have the approximation for the white-noise case:

$$\mathrm{Cov}(\boldsymbol{x}^*) = \varsigma^2 (A^TA)^{-1} \simeq \frac{\varsigma^2}{m} F^{-1}.$$

We see that the solution's variance is (to a good approximation) inversely proportional to the number m of data points.

To illustrate the above result we consider again the frozen cod meat example, this time with two sets of abscissas t_i uniformly distributed in $[0, 0.4]$ for $m = 50$ and $m = 200$, leading to the two matrices $(A^TA)^{-1}$

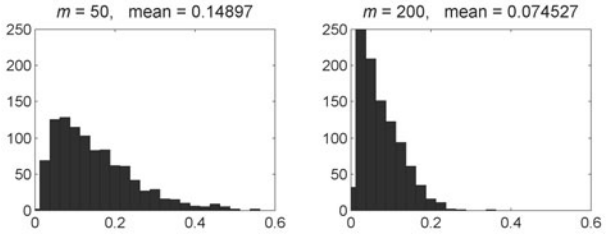

Figure 2.1.3: Histograms of the error norms $\|\boldsymbol{x}^{\text{exact}} - \boldsymbol{x}^*\|_2$ for the two test problems with additive white noise; the errors are clearly reduced by a factor of 2 when we increase m from 50 to 200.

given by

$$\begin{pmatrix} 1.846 & -1.300 & 0.209 \\ -1.300 & 1.200 & -0.234 \\ 0.209 & -0.234 & 0.070 \end{pmatrix} \quad \textit{and} \quad \begin{pmatrix} 0.535 & -0.359 & 0.057 \\ -0.359 & 0.315 & -0.061 \\ 0.057 & -0.061 & 0.018 \end{pmatrix},$$

respectively. The average ratio between the elements in the two matrices is 3.71*, i.e., fairly close to the factor* 4 *we expect from the above analysis when increasing* m *by a factor* 4*.*

We also solved the two LSQ problems for 1000 realizations of additive white noise, and Figure 2.1.3 shows histograms of the error norms $\|\boldsymbol{x}^{\text{exact}} - \boldsymbol{x}^*\|_2$*, where* $\boldsymbol{x}^{\text{exact}} = (1.27,\, 2.04,\, 0.3)^T$ *is the vector of exact parameters for the problem. These results confirm that the errors are reduced by a factor of* 2 *corresponding to the expected reduction of the standard deviation by the same factor.*

2.2 The QR factorization and its role

In this and the next section we discuss the QR factorization and its role in the analysis and solution of the LSQ problem. We start with the simpler case of full-rank matrices in this section and then move on to rank-deficient matrices in the next section.

The first step in the computation of a solution to the least squares problem is the reduction of the problem to an equivalent one with a more convenient matrix structure. This can be done through an explicit factorization, usually based on orthogonal transformations, where instead of solving the original LSQ problem (2.1.1) one solves an equivalent problem with a triangular matrix. The basis of this procedure is the QR factorization, the less expensive decomposition that takes advantage of the isometric properties of orthogonal transformations (proofs for all the theorems in this section can be found in [20], [105] and many other references).

Theorem 16. ***QR factorization.*** *Any real $m \times n$ matrix A can be factored as*

$$\boxed{A = QR \quad with \quad Q \in \mathbb{R}^{m\times m}, \quad R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \in \mathbb{R}^{m\times n},} \tag{2.2.1}$$

where Q is orthogonal (i.e., $Q^TQ = I_m$) and $R_1 \in \mathbb{R}^{n\times n}$ is upper triangular. If A has full rank, then so has R and therefore all its diagonal elements are nonzero.

Theorem 17. ***Economical QR factorization.*** *Let $A \in \mathbb{R}^{m\times n}$ have full column rank $r = n$. The economical (or thin) QR factorization of A is*

$$\boxed{A = Q_1R_1 \quad with \quad Q_1 \in \mathbb{R}^{m\times n}, \quad R_1 \in \mathbb{R}^{n\times n},} \tag{2.2.2}$$

where Q_1 has orthonormal columns (i.e., $Q_1^TQ_1 = I_n$) and the upper triangular matrix R_1 has nonzero diagonal entries. Moreover, Q_1 can be chosen such that the diagonal elements of R_1 are positive, in which case R_1 is the Cholesky factor of A^TA.

Similar theorems hold if the matrix A is complex, with the factor Q now a unitary matrix.

Remark 18. *If we partition the $m \times m$ matrix Q in the full QR factorization* (2.2.1) *as*

$$Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix},$$

then the sub-matrix Q_1 is the one that appears in the economical QR factorization (2.2.2)*. The $m \times (m - n)$ matrix Q_2 satisfies $Q_2^TQ_1 = 0$ and $Q_1Q_1^T + Q_2Q_2^T = I_m$.*

Geometrically, the QR factorization corresponds to an orthogonalization of the linearly independent columns of A. The columns of matrix Q_1 are an orthonormal basis for range(A) and those of Q_2 are an orthonormal basis for null(A^T).

The following theorem expresses the least squares solution of the full-rank problem in terms of the economical QR factorization.

Theorem 19. *Let $A \in \mathbb{R}^{m\times n}$ have full column rank $r = n$, with the economical QR factorization $A = Q_1R_1$ from Theorem 17. Considering that*

$$\begin{aligned} \|\boldsymbol{b} - A\boldsymbol{x}\|_2^2 &= \|Q^T(\boldsymbol{b} - A\boldsymbol{x})\|_2^2 = \| \begin{pmatrix} Q_1^T\boldsymbol{b} \\ Q_2^T\boldsymbol{b} \end{pmatrix} - \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \boldsymbol{x}\|_2^2 \\ &= \|Q_1^T\boldsymbol{b} - R_1\boldsymbol{x}\|_2^2 + \|Q_2^T\boldsymbol{b}\|_2^2, \end{aligned}$$

then, the unique solution of the LSQ problem $\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2$ *can be computed from the simpler, equivalent problem*

$$\min_{\boldsymbol{x}} \|Q_1^T\boldsymbol{b} - R_1\boldsymbol{x}\|_2^2,$$

whose solution is

$$\boldsymbol{x}^* = R_1^{-1}Q_1^T\boldsymbol{b} \tag{2.2.3}$$

and the corresponding least squares residual is given by

$$\boldsymbol{r}^* = \boldsymbol{b} - A\,\boldsymbol{x}^* = (I_m - Q_1Q_1^T)\boldsymbol{b} = Q_2Q_2^T\boldsymbol{b}, \tag{2.2.4}$$

with the matrix Q_2 *that was introduced in Remark 18.*

Of course, (2.2.3) is short-hand for solving $R\,x^* = Q_1^T\boldsymbol{b}$, and one point of this reduction is that it is much simpler to solve a triangular system of equations than a full one. Further on we will also see that this approach has better numerical properties, as compared to solving the normal equations introduced in the previous section.

Example 20. *In Example 11 we saw the normal equations for the NMR problem from Example 1; here we take a look at the economical QR factorization for the same problem:*

$$A = \begin{pmatrix} 1.00 & 1.00 & 1 \\ 0.80 & 0.94 & 1 \\ 0.64 & 0.88 & 1 \\ \vdots & \vdots & \vdots \\ 3.2\cdot 10^{-5} & 4.6\cdot 10^{-2} & 1 \\ 2.5\cdot 10^{-5} & 4.4\cdot 10^{-2} & 1 \\ 2.0\cdot 10^{-5} & 4.1\cdot 10^{-2} & 1 \end{pmatrix},$$

$$Q_1 = \begin{pmatrix} 0.597 & -0.281 & 0.172 \\ 0.479 & -0.139 & 0.071 \\ 0.384 & -0.029 & -0.002 \\ \vdots & \vdots & \vdots \\ 1.89\cdot 10^{-5} & 0.030 & 0.224 \\ 1.52\cdot 10^{-5} & 0.028 & 0.226 \\ 1.22\cdot 10^{-5} & 0.026 & 0.229 \end{pmatrix},$$

$$R_1 = \begin{pmatrix} 1.67 & 2.40 & 3.02 \\ 0 & 1.54 & 5.16 \\ 0 & 0 & 3.78 \end{pmatrix},\ Q\boldsymbol{b} = \begin{pmatrix} 7.81 \\ 4.32 \\ 1.19 \end{pmatrix}.$$

We note that the upper triangular matrix R_1 *is also the Cholesky factor of the normal equation matrix, i.e.,* $A^TA = R_1^TR_1$.

The QR factorization allows us to study the residual vector in more detail. Consider first the case where we augment A with an additional column, corresponding to adding an additional model basis function in the data fitting problem.

Theorem 21. *Let the augmented matrix* $\overline{A} = (\, A \,,\, \boldsymbol{a}_{n+1} \,)$ *have the QR factorization*

$$\overline{A} = \begin{pmatrix} \overline{Q}_1 & \overline{Q}_2 \end{pmatrix} \begin{pmatrix} \overline{R}_1 \\ 0 \end{pmatrix},$$

with $\overline{Q}_1 = (\; Q_1 \quad \bar{\boldsymbol{q}} \;)$, $Q_1^T \bar{\boldsymbol{q}} = \boldsymbol{0}$ *and* $\overline{Q}_2^T \bar{\boldsymbol{q}} = \boldsymbol{0}$. *Then the norms of the least squares residual vectors* $\boldsymbol{r}^* = (I_m - Q_1 Q_1^T)\boldsymbol{b}$ *and* $\bar{\boldsymbol{r}}^* = (I_m - \overline{Q}_1 \overline{Q}_1^T)\boldsymbol{b}$ *are related by*

$$\|\boldsymbol{r}^*\|_2^2 = \|\bar{\boldsymbol{r}}^*\|_2^2 + (\bar{\boldsymbol{q}}^T \boldsymbol{b})^2.$$

Proof. From the relation $\overline{Q}_1 \overline{Q}_1^T = Q_1 Q_1^T + \bar{\boldsymbol{q}} \bar{\boldsymbol{q}}^T$ it follows that $I_m - Q_1 Q_1^T = I_m - \overline{Q}_1 \overline{Q}_1^T + \bar{\boldsymbol{q}} \bar{\boldsymbol{q}}^T$, and hence,

$$\begin{aligned} \|\boldsymbol{r}^*\|_2^2 &= \|(I_m - Q_1 Q_1^T)\boldsymbol{b}\|_2^2 = \|(I_m - \overline{Q}_1 \overline{Q}_1^T)\boldsymbol{b} + \bar{\boldsymbol{q}} \bar{\boldsymbol{q}}^T \boldsymbol{b}\|_2^2 \\ &= \|(I_m - \overline{Q}_1 \overline{Q}_1^T)\boldsymbol{b}\|_2^2 + \|\bar{\boldsymbol{q}} \bar{\boldsymbol{q}}^T \boldsymbol{b}\|_2^2 = \|\bar{\boldsymbol{r}}^*\|_2^2 + (\bar{\boldsymbol{q}}^T \boldsymbol{b})^2, \end{aligned}$$

where we used that the two components of $\boldsymbol{r}^*$ are orthogonal and that $\|\bar{\boldsymbol{q}} \bar{\boldsymbol{q}}^T \boldsymbol{b}\|_2 = |\bar{\boldsymbol{q}}^T \boldsymbol{b}| \, \|\bar{\boldsymbol{q}}\|_2 = |\bar{\boldsymbol{q}}^T \boldsymbol{b}|$. □

This theorem shows that, when we increase the number of model basis functions for the fit in such a way that the matrix retains full rank, then the least squares residual norm decreases (or stays fixed if $\boldsymbol{b}$ is orthogonal to $\bar{\boldsymbol{q}}$).

To obtain more insight into the least squares residual we study the influence of the approximation and data errors. According to (1.2.1) we can write the right-hand side as

$$\boldsymbol{b} = \boldsymbol{\Gamma} + \boldsymbol{e},$$

where the two vectors

$$\boldsymbol{\Gamma} = \big(\Gamma(t_1), \ldots, \Gamma(t_m)\big)^T \qquad \text{and} \qquad \boldsymbol{e} = (e_1, \ldots, e_m)^T$$

contain the pure data (the sampled pure-data function) and the data errors, respectively. Hence, the least squares residual vector is

$$\boldsymbol{r}^* = \boldsymbol{\Gamma} - A\,\boldsymbol{x}^* + \boldsymbol{e}, \tag{2.2.5}$$

where the vector $\boldsymbol{\Gamma} - A\,\boldsymbol{x}^*$ is the approximation error. From (2.2.5) it follows that the least squares residual vector can be written as

$$\boldsymbol{r}^* = (I_m - Q_1 Q_1^T)\,\boldsymbol{\Gamma} + (I_m - Q_1 Q_1^T)\,\boldsymbol{e} = Q_2 Q_2^T \boldsymbol{\Gamma} + Q_2 Q_2^T \boldsymbol{e}.$$

We see that the residual vector consists of two terms. The first term $Q_2Q_2^T\mathbf{\Gamma}$ is an "approximation residual," due to the discrepancy between the n model basis functions (represented by the columns of A) and the pure-data function. The second term is the "projected error", i.e., the component of the data errors that lies in the subspace null(A^T). We can summarize the statistical properties of the least squares residual vector as follows.

Theorem 22. *The least squares residual vector* $\boldsymbol{r}^* = \boldsymbol{b} - A\,\boldsymbol{x}^*$ *has the following properties:*

$$\mathcal{E}(\boldsymbol{r}^*) = Q_2Q_2^T\mathbf{\Gamma}, \quad \mathrm{Cov}(\boldsymbol{r}^*) = Q_2Q_2^T\mathrm{Cov}(\boldsymbol{e})Q_2Q_2^T,$$

$$\mathcal{E}(\|\boldsymbol{r}^*\|_2^2) = \|Q_2^T\mathbf{\Gamma}\|_2^2 + \mathcal{E}(\|Q_2^T\boldsymbol{e}\|_2^2).$$

If $\boldsymbol{e}$ *is white noise, i.e.,* $\mathrm{Cov}(\boldsymbol{e}) = \varsigma^2 I_m$*, then*

$$\mathrm{Cov}(\boldsymbol{r}^*) = \varsigma^2 Q_2\,Q_2^T, \quad E(\|\boldsymbol{r}^*\|_2^2) = \|Q_2^T\mathbf{\Gamma}\|_2^2 + (m-n)\,\varsigma^2.$$

Proof. It follows immediately that

$$\mathcal{E}(Q_2Q_2^T\boldsymbol{e}) = Q_2Q_2^T\mathcal{E}(\boldsymbol{e}) = 0 \ \text{ and } \ E(\mathbf{\Gamma}^TQ_2Q_2^T\boldsymbol{e}) = 0,$$

as well as

$$\mathrm{Cov}(\boldsymbol{r}^*) = Q_2Q_2^T\mathrm{Cov}(\mathbf{\Gamma}+\boldsymbol{e})Q_2Q_2^T \ \text{ and } \ \mathrm{Cov}(\mathbf{\Gamma}+\boldsymbol{e}) = \mathrm{Cov}(\boldsymbol{e}).$$

Moreover,

$$\mathcal{E}(\|\boldsymbol{r}^*\|_2^2) = \mathcal{E}(\|Q_2Q_2^T\mathbf{\Gamma}\|_2^2) + \mathcal{E}(\|Q_2Q_2^T\boldsymbol{e}\|_2^2) + \mathcal{E}(2\mathbf{\Gamma}^TQ_2Q_2^T\boldsymbol{e}).$$

It follows that

$$\mathrm{Cov}(Q_2^T\boldsymbol{e}) = \varsigma^2 I_{m-n} \text{ and } \mathcal{E}(\|Q_2^T\boldsymbol{e}\|_2^2) = \mathrm{trace}(\mathrm{Cov}(Q_2^T\boldsymbol{e})) = (m-n)\varsigma^2.$$

□

From the above theorem we see that if the approximation error $\mathbf{\Gamma} - A\,\boldsymbol{x}^*$ is somewhat smaller than the data error $\boldsymbol{e}$ then, in the case of white noise, the scaled residual norm s^* (sometimes referred to as the standard error), defined by

$$s^* = \frac{\|\boldsymbol{r}^*\|_2}{\sqrt{m-n}}, \tag{2.2.6}$$

provides an estimate for the standard deviation ς of the errors in the data. Moreover, provided that the approximation error decreases sufficiently fast when the fitting order n increases, then we should expect that for large

enough n the least squares residual norm becomes dominated by the projected error term, i.e.,

$$\boldsymbol{r}^* \simeq Q_2 Q_2^T \boldsymbol{e} \qquad \text{for } n \text{ sufficiently large.}$$

Hence, if we monitor the scaled residual norm $s^* = s^*(n)$ as a function of n, then we expect to see that $s^*(n)$ initially decreases – when it is dominated by the approximation error – while at a later stage it levels off, when the projected data error dominates. The transition between the two stages of the behavior of $s^*(n)$ indicates a good choice for the fitting order n.

Example 23. *We return to the air pollution example from Example 2. We compute the polynomial fit for* $n = 1, 2, \ldots, 19$ *and the trigonometric fit for* $n = 1, 3, 5, \ldots, 19$ *(only odd values of* n *are used, because we always need a* sin-cos *pair). Figure 2.2.1 shows the residual norm* $\|\boldsymbol{r}^*\|_2$ *and the scaled residual norm* s^* *as functions of* n*.*

The residual norm decreases monotonically with n*, while the scaled residual norm shows the expected behavior mentioned above, i.e., a decaying phase (when the approximation error dominates), followed by a more flat or slightly increasing phase when the data errors dominate.*

The standard error s^* introduced in (2.2.6) above, defined as the residual norm adjusted by the degrees of freedom in the residual, is just one example of a quantity from statistics that plays a central role in the analysis of LSQ problems. Another quantity arising from statistics is the *coefficient of determination* R^2, which is used in the context of linear regression analysis (statistical modeling) as a measure of how well a linear model fits the data. Given a model $M(\boldsymbol{x}, t)$ that predicts the observations $b_1, b_2, \ldots, b_m$ and the residual vector $\boldsymbol{r} = (b_1 - M(\boldsymbol{x}, t_1), \ldots, b_m - M(\boldsymbol{x}, t_m))^T$, the coefficient of determination is defined by

$$R^2 = 1 - \frac{\|\boldsymbol{r}\|_2^2}{\sum_{i=1}^m (b_i - \bar{b})^2}, \tag{2.2.7}$$

where $\bar{b}$ is the mean of the observations. In general, it is an approximation of the unexplained variance, since the second term compares the variance in the model's errors with the total variance of the data. Yet another useful quantity for analysis is the *adjusted coefficient of determination*, $\operatorname{adj} R^2$, defined in the same way as the coefficient of determination R^2, but adjusted using the residual degrees of freedom,

$$\operatorname{adj} R^2 = 1 - \frac{(s^*)^2}{\sum_{i=1}^m (b_i - \bar{b})^2/(m-1)}, \tag{2.2.8}$$

making it similar in spirit to the squared standard error $(s^*)^2$. In Chapter 11 we demonstrate the use of these statistical tools.

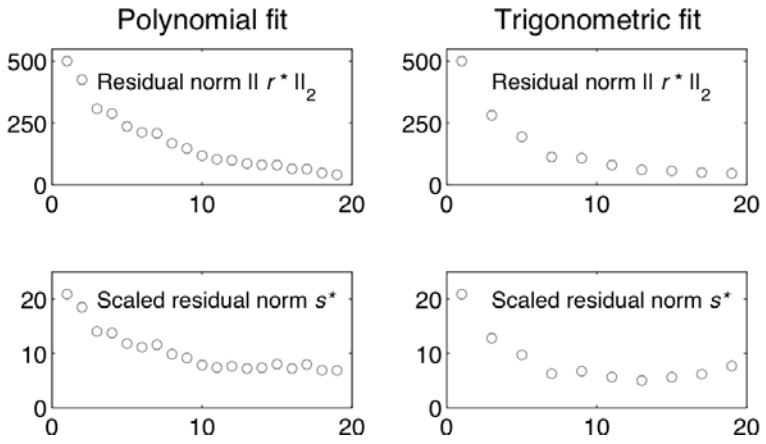

Figure 2.2.1: The residual norm and the scaled residual norm, as functions of the fitting order n, for the polynomial and trigonometric fits to the air pollution data.

2.3 Permuted QR factorization

The previous section covered in detail full-rank problems, and we saw that the QR factorization was well suited for solving such problems. However, for parameter estimation problems – where the model is given – there is no guarantee that A always has full rank, and therefore we must also consider the rank-deficient case. We give first an overview of some matrix factorizations that are useful for detecting and treating rank-deficient problems, although they are of course also applicable in the full-rank case. The minimum-norm solution from Definition 27 below plays a central role in this discussion.

When A is rank deficient we cannot always compute a QR factorization (2.2.1) that has a convenient economical version, where the range of A is spanned by the first columns of Q. The following example illustrates that a column permutation is needed to achieve such a form.

Example 24. *Consider the factorization*

$$A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} 0 & s \\ 0 & c \end{pmatrix}, \qquad \textit{for any } c^2 + s^2 = 1.$$

This QR factorization has the required form, i.e., the first factor is orthogonal and the second is upper triangular – but range(A) *is not spanned by the first column of the orthogonal factor. However, a permutation* Π *of the*

columns of A gives a QR factorization of the desired form,

$$A\,\Pi = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = Q\,R = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

with a triangular R and such that the range of A is spanned by the first column of Q.

In general, we need a permutation of columns that selects the linearly independent columns of A and places them first. The following theorem formalizes this idea.

Theorem 25. ***QR factorization with column permutation.*** *If A is real, $m \times n$ with $\operatorname{rank}(A) = r < n \leq m$, then there exists a permutation Π, not necessarily unique, and an orthogonal matrix Q such that*

$$A\,\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{matrix} r \\ m-r \end{matrix}, \tag{2.3.1}$$

where R_{11} is $r \times r$ upper triangular with positive diagonal elements. The range of A is spanned by the first r columns of Q.

Similar results hold for complex matrices where Q now is unitary.

The first r columns of the matrix $A\,\Pi$ are guaranteed to be linearly independent. For a model with basis functions that are not linearly dependent over the abscissas, this provides a method for choosing r linearly independent functions. The rank-deficient least squares problem can now be solved as follows.

Theorem 26. *Let A be a rank-deficient $m \times n$ matrix with the pivoted QR factorization in Theorem 25. Then the LSQ problem* (2.1.1) *takes the form*

$$\begin{aligned} \min_x \|Q^T A\,\Pi\,\Pi^T \boldsymbol{x} - Q^T b\|_2^2 \quad &= \\ \min_y \left\| \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix} - \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{pmatrix} \right\|_2^2 \quad &= \\ \min_y \left(\|R_{11}\boldsymbol{y}_1 - R_{12}\boldsymbol{y}_2 - \boldsymbol{d}_1\|_2^2 + \|\boldsymbol{d}_2\|_2^2 \right), & \end{aligned}$$

where we have introduced

$$Q^T \boldsymbol{b} = \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{y} = \Pi^T \boldsymbol{x} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix}.$$

The general solution is

$$\boldsymbol{x}^* = \Pi \begin{pmatrix} R_{11}^{-1} \left(\boldsymbol{d}_1 - R_{12}\,\boldsymbol{y}_2 \right) \\ \boldsymbol{y}_2 \end{pmatrix}, \qquad \boldsymbol{y}_2 = \text{arbitrary} \tag{2.3.2}$$

and any choice of $\boldsymbol{y}_2$ leads to a least squares solution with residual norm $\|\boldsymbol{r}^*\|_2 = \|\boldsymbol{d}_2\|_2$.

Definition 27. *Given the LSQ problem with a rank deficient matrix A and the general solution given by (2.3.2), we define $\widehat{\boldsymbol{x}}^*$ as the solution of minimal l_2-norm that satisfies*

$$\widehat{\boldsymbol{x}}^* = \arg\,\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2 \qquad \textit{subject to} \qquad \|\boldsymbol{b} - A\boldsymbol{x}\|_2 = \min\,.$$

The choice $\boldsymbol{y}_2 = \boldsymbol{0}$ in (2.3.2) is an important special case that leads to the so-called *basic solution*,

$$\boldsymbol{x}_{\mathrm{B}} = \Pi \begin{pmatrix} R_{11}^{-1} Q_1^T \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix},$$

with at least $n-r$ zero components. This corresponds to using only the first r columns of $A\,\Pi$ in the solution, while setting the remaining elements to zero. As already mentioned, this is an important choice in data fitting – as well as other applications – because it implies that $\boldsymbol{b}$ is represented by the smallest subset of r columns of A, i.e., it is fitted with as few variables as possible. It is also related to the new field of compressed sensing [4, 35, 235].

Example 28. ***Linear prediction.*** *We consider a digital signal, i.e., a vector $\boldsymbol{s} \in \mathbb{R}^N$, and we seek a relation between neighboring elements of the form*

$$s_i = \sum_{j=1}^{\ell} x_i\, s_{i-j}, \qquad i = p+1, \ldots, N, \tag{2.3.3}$$

for some (small) value of ℓ. The technique of estimating the ith element from a number of previous elements is called linear prediction (LP), and the LP coefficients x_i can be used to characterize various underlying properties of the signal. Throughout this book we will use a test problem where the elements of the noise-free signal are given by

$$s_i = \alpha_1\, \sin(\omega_1\, t_i) + \alpha_2\, \sin(\omega_2\, t_i) + \cdots + \alpha_p\, \sin(\omega_p\, t_i), \quad i = 1, 2, \ldots, N.$$

In this particular example, we use $N = 32$, $p = 2$, $\alpha_1 = 2$, $\alpha_2 = -1$ and no noise.

There are many ways to estimate the LP coefficients in (2.3.3)*. One of the popular methods amounts to forming a Toeplitz matrix A (i.e., a matrix with constant diagonals) and a right-hand side $\boldsymbol{b}$ from the signal, with elements given by*

$$a_{ij} = s_{n+i-j}, \quad b_i = s_{n+i}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n,$$

where the matrix dimensions m and n satisfy $m+n = N$ and $\min(m, n) \geq \ell$. We choose $N = 32$ and $n = 7$ giving $m = 25$, and the first 7 rows of A and

the first 7 elements of $\boldsymbol{b}$ *are*

$$\begin{pmatrix} 1.011 & -1.151 & -0.918 & -2.099 & -0.029 & 2.770 & 0.875 \\ 2.928 & 1.011 & -1.151 & -0.918 & -2.099 & -0.029 & 2.770 \\ 1.056 & 2.928 & 1.011 & -1.151 & -0.918 & -2.099 & -0.029 \\ -1.079 & 1.056 & 2.928 & 1.011 & -1.151 & -0.918 & -2.099 \\ -1.197 & -1.079 & 1.056 & 2.928 & 1.011 & -1.151 & -0.918 \\ -2.027 & -1.197 & -1.079 & 1.056 & 2.928 & 1.011 & -1.151 \\ 1.160 & -2.027 & -1.197 & -1.079 & 1.056 & 2.928 & 1.011 \end{pmatrix}, \quad \begin{pmatrix} 2.928 \\ 0.0101 \\ -1.079 \\ -1.197 \\ -2.027 \\ 1.160 \\ 2.559 \end{pmatrix}.$$

The matrix A *is rank deficient and it turns out that for this problem we can safely compute the ordinary QR factorization without pivoting, corresponding to* $\Pi = I$*. The matrix* R_1 *and the vector* $Q_1^T\boldsymbol{b}$ *are*

$$\begin{pmatrix} 7.970 & 2.427 & -3.392 & -4.781 & -5.273 & 1.890 & 7.510 \\ 0 & 7.678 & 3.725 & -1.700 & -3.289 & -6.482 & -0.542 \\ 0 & 0 & 6.041 & 2.360 & -4.530 & -1.136 & -2.765 \\ 0 & 0 & 0 & 5.836 & -0.563 & -4.195 & 0.252 \\ 0 & 0 & 0 & 0 & \epsilon & \epsilon & \epsilon \\ 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon \\ 0 & 0 & 0 & 0 & 0 & 0 & \epsilon \end{pmatrix}, \quad \begin{pmatrix} 2.573 \\ -4.250 \\ -1.942 \\ -5.836 \\ \epsilon \\ \epsilon \\ \epsilon \end{pmatrix},$$

where ϵ *denotes an element whose absolute value is of the order* 10^{-14} *or smaller. We see that the numerical rank of* A *is* $r = 4$ *and that* $\boldsymbol{b}$ *is the weighted sum of columns 1 through 4 of the matrix* A*, i.e., four LP coefficients are needed in* (2.3.3)*. A basic solution is obtained by setting the last three elements of the solution to zero:*

$$\boldsymbol{x}_{\mathrm{B}} = (\,-0.096\,,\,-0.728\,,\,-0.096\,,\,-1.000\,,\,0\,,\,0\,,\,0\,)^T.$$

A numerically safer approach for rank-deficient problems is to use the QR factorization with column permutations from Theorem 25, for which we get

$$\Pi = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and

$$R_{11} = \begin{pmatrix} 8.052 & 1.747 & -3.062 & -4.725 \\ 0 & 7.833 & -4.076 & -2.194 \\ 0 & 0 & 5.972 & -3.434 \\ 0 & 0 & 0 & 5.675 \end{pmatrix}, \quad \boldsymbol{d}_1 = \begin{pmatrix} -3.277 \\ 3.990 \\ -5.952 \\ 6.79 \end{pmatrix}.$$

The basic solution corresponding to this factorization is

$$\boldsymbol{x}_{\mathrm{B}} = (\,0\,,\,-0.721\,,\,0\,,\,-0.990\,,\,0.115\,,\,0\,,\,0.027\,)^T.$$

This basic solution expresses $\boldsymbol{b}$ *as a weighted sum of columns 1, 3, 4 and 6 of* A*. The example shows that the basic solution is not unique – both basic solutions given above solve the LSQ problem associated with the linear prediction problem.*

The basic solution introduced above is one way of defining a particular type of solution of the rank-deficient LSQ problem, and it is useful in some applications. However, in other applications we require the minimum-norm solution $\widehat{\boldsymbol{x}}^*$ from Definition 27, whose computation reduces to solving the least squares problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2 = \min_{\boldsymbol{y}_2} \left\| \Pi \begin{pmatrix} R_{11}^{-1}(\boldsymbol{d}_1 - R_{12}\boldsymbol{y}_2) \\ \boldsymbol{y}_2 \end{pmatrix} \right\|_2 .$$

Using the basic solution $\boldsymbol{x}_{\mathrm{B}}$ the problem is reduced to

$$\min_{\boldsymbol{y}_2} \left\| \boldsymbol{x}_{\mathrm{B}} - \Pi \begin{pmatrix} R_{11}^{-1}R_{12} \\ I \end{pmatrix} \boldsymbol{y}_2 \right\|_2 .$$

This is a full-rank least squares problem with matrix $\Pi \begin{pmatrix} R_{11}^{-1}R_{12} \\ I \end{pmatrix}$. The right-hand side $\boldsymbol{x}_{\mathrm{B}}$ and the solution $\boldsymbol{y}_2^*$ can be obtained via a QR factorization. The following results from [101] relates the norms of the basic and minimum-norm solutions:

$$1 \le \frac{\|\boldsymbol{x}_{\mathrm{B}}\|_2}{\|\widehat{\boldsymbol{x}}^*\|_2} \le \sqrt{1 + \|R_{11}^{-1}R_{12}\|_2^2}.$$

Complete orthogonal factorization

As demonstrated above, we cannot immediately compute the minimum-norm least squares solution $\widehat{\boldsymbol{x}}^*$ from the pivoted QR factorization. However, the QR factorization with column permutations can be considered as a first step toward the so-called *complete orthogonal factorization*. This is a decomposition that, through basis changes by means of orthogonal transformations in both $\mathbb{R}^m$ and $\mathbb{R}^n$, concentrates the whole information of A into a leading square nonsingular matrix of size $r \times r$. This gives a more direct way of computing $\widehat{\boldsymbol{x}}^*$. The existence of complete orthogonal factorizations is stated in the following theorem.

Theorem 29. *Let* A *be a real* $m \times n$ *matrix of rank* r*. Then there is an* $m \times m$ *orthogonal matrix* U *and an* $n \times n$ *orthogonal matrix* V *such that*

$$A = URV^T \quad \textit{with} \quad R = \begin{pmatrix} R_{11} & 0 \\ 0 & 0 \end{pmatrix}, \tag{2.3.4}$$

where R_{11} is an $r \times r$ nonsingular triangular matrix.

A similar result holds for complex matrices with U and V unitary. The LSQ solution can now be obtained as stated in the following theorem.

Theorem 30. *Let A have the complete orthogonal decomposition* (2.3.4) *and introduce the auxiliary vectors*

$$U^T\boldsymbol{b} = \boldsymbol{g} = \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \end{pmatrix} \begin{matrix} r \\ m-r \end{matrix}, \quad V^T\boldsymbol{x} = \boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix} \begin{matrix} r \\ n-r \end{matrix}. \tag{2.3.5}$$

Then the solutions to $\min_x \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2$ are given by

$$\boldsymbol{x}^* = V \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix}, \qquad \boldsymbol{y}_1 = R_{11}^{-1}\boldsymbol{g}_1, \qquad \boldsymbol{y}_2 = \textit{arbitrary}, \tag{2.3.6}$$

and the residual norm is $\|\boldsymbol{r}^*\|_2 = \|\boldsymbol{g}_2\|_2$. In particular, the minimum-norm solution $\widehat{\boldsymbol{x}}^*$ is obtained by setting $\boldsymbol{y}_2 = \boldsymbol{0}$.

Proof. Replacing A by its complete orthogonal decomposition we get

$$\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2 = \|\boldsymbol{b} - URV^T\boldsymbol{x}\|_2^2 = \|U^T\boldsymbol{b} - RV^T\boldsymbol{x}\|_2^2 = \|\boldsymbol{g}_1 - R_{11}\boldsymbol{y}_1\|_2^2 + \|\boldsymbol{g}_2\|_2^2.$$

Since the sub-vector $\boldsymbol{y}_2$ cannot lower this minimum, it can be chosen arbitrarily and the result follows. □

The triangular matrix R_{11} contains all the fundamental information of A. The SVD, which we will introduce shortly, is a special case of a complete orthogonal factorization, which is more computationally demanding and involves an iterative part. The most sparse structure that can be obtained by a finite number of orthogonal transformations, the bidiagonal case, is left to be analyzed exhaustively in the chapter on direct numerical methods.

Example 31. *We return to the linear prediction example from the previous section; this time we compute the complete orthogonal factorization from Theorem 29 and get*

$$R_{11} = \begin{pmatrix} -9.027 & -4.690 & -1.193 & 5.626 \\ 0 & -8.186 & -3.923 & -0.373 \\ 0 & 0 & 9.749 & 5.391 \\ 0 & 0 & 0 & 9.789 \end{pmatrix}, \quad \boldsymbol{g} = \begin{pmatrix} -1.640 \\ 3.792 \\ -6.704 \\ 0.712 \end{pmatrix},$$

and

$$V = \begin{pmatrix} -0.035 & 0.355 & -0.109 & -0.521 & -0.113 & -0.417 & 0.634 \\ 0.582 & -0.005 & 0.501 & -0.103 & 0.310 & -0.498 & -0.237 \\ 0.078 & 0.548 & 0.044 & 0.534 & 0.507 & 0.172 & 0.347 \\ -0.809 & 0.034 & 0.369 & 0.001 & 0.277 & -0.324 & -0.163 \\ 0 & -0.757 & -0.006 & 0.142 & 0.325 & -0.083 & 0.543 \\ 0 & 0 & -0.774 & 0.037 & 0.375 & -0.408 & -0.305 \\ 0 & 0 & 0 & -0.641 & 0.558 & 0.520 & -0.086 \end{pmatrix}.$$

This factorization is not unique and the zeros in V are due to the particular algorithm from [78] used here. The minimum-norm solution is

$$\begin{aligned} \widehat{\boldsymbol{x}}^* &= V \begin{pmatrix} R_{11}^{-1}\boldsymbol{g}_1 \\ \mathbf{0} \end{pmatrix} \\ &= (\,-0.013\,,\,-0.150\,,\,-0.028\,,\,-0.581\,,\,0.104\,,\,0.566\,,\,-0.047\,)^T. \end{aligned}$$

This solution, as well as the basic solutions from the previous example, all solve the rank-deficient least squares problem.

Example 32. *We will show yet another way to compute the linear prediction coefficients that uses the null space information in the complete orthogonal factorization. In particular, we observe that the last three columns of V span the null space of A. If we extract them to form the matrix V_0 and compute a QR factorization of its transpose, then we get*

$$V_0^T = Q_0 R_0, \qquad R_0^T = \begin{pmatrix} 0.767 & 0 & 0 \\ 0.031 & 0.631 & 0 \\ 0.119 & -0.022 & 0.626 \\ 0.001 & 0.452 & 0.060 \\ 0.446 & 0.001 & 0.456 \\ -0.085 & 0.624 & 0.060 \\ -0.436 & -0.083 & 0.626 \end{pmatrix}.$$

Since $A\,R_0^T = 0$, we can normalize the last column (by dividing it by its maximum norm) to obtain the null vector

$$\boldsymbol{v}_0 = (\,0\,,\,0\,,\,1\,,\,0.096\,,\,0.728\,,\,0.096\,,\,1\,)^T,$$

which is another way to describe the linear dependence between five neighboring columns $\boldsymbol{a}_j$ of A. Specifically, this $\boldsymbol{v}_0$ states that $\boldsymbol{a}_7 = -0.096\boldsymbol{a}_6 - 0.728\boldsymbol{a}_5 - 0.96\boldsymbol{a}_4 - \boldsymbol{a}_3$, and it follows that the LP coefficients are $x_1 = -0.728$, $x_2 = -0.096$, $x_3 = -0.096$ and $x_4 = -1$, which is identical to the results in Example 28.

This page intentionally left blank

Chapter 3

Analysis of Least Squares Problems

The previous chapter has set the stage for introducing some fundamental tools for analyzing further the LSQ problem, namely, the pseudoinverse and the singular value decomposition. After introducing these concepts, we complete this chapter with a definition of the condition number for least squares problems, followed by a discussion of the robustness of the solution to perturbations in the data or the model.

3.1 The pseudoinverse

The previous chapter described computational expressions for computing LSQ solutions to full-rank and rank-deficient problems. However, it is also convenient to have a simple, closed form expression for the LSQ solution, similar to the notation $\boldsymbol{x} = A^{-1}\boldsymbol{b}$ for full-rank square problems.

The goal of this section is to define the pseudoinverse of a rectangular matrix that plays a similar role to the inverse of a square nonsingular matrix, but has only some of its properties. The concept of generalized inverse found in the literature varies. We will use the following definition, appropriate for the least squares problem.

Definition 33. ***Pseudoinverse.*** *Corresponding to any real rectangular $m \times n$ matrix A there exists a unique $n \times m$ matrix X having the following four properties:*

$$(i)\ AXA = A, \quad (ii)\ XAX = X,$$

$$(iii)\ (AX)^T = AX, \quad (iv)\ (XA)^T = XA.$$

For a complex matrix, the last two conditions are $(AX)^H = AX$ *and* $(XA)^H = XA$*, where* H *stands for conjugate transpose. This matrix* X *is called the pseudoinverse (or Moore-Penrose inverse) and is denoted by* $A^\dagger$*. A matrix* X *that satisfies conditions (i)–(iii) is called a generalized inverse* A^-*; if* A *is rank deficient, then* A^- *is not unique.*

The pseudoinverse satisfies the relations $(A^\dagger)^\dagger = A$ and $(A^T)^\dagger = (A^\dagger)^T$ (or $(A^H)^\dagger = (A^\dagger)^H$ in the complex case), but we note that there are important "inverse properties" that the pseudoinverse **does not have** in general, such as $(AB)^\dagger \neq (BA)^\dagger$ and $A\,A^\dagger \neq A^\dagger A$. In several special cases the pseudoinverse of A has an expression in terms of the inverse of some matrix:

- If $r = n = m$, then $A^\dagger = A^{-1}$.
- If $r = n < m$ and A is real, then $A^\dagger = (A^T A)^{-1} A^T$.
- If both $A \in \mathbb{R}^{m\times r}$ and $B \in \mathbb{R}^{r\times n}$ have full rank $r \leq \min(m, n)$, then

$$(AB)^\dagger = B^\dagger A^\dagger = B^T (BB^T)^{-1} (A^T A)^{-1} A^T.$$

- The pseudoinverse has a simple expression in terms of the complete orthogonal decomposition of Theorem 29:

$$A = U \begin{pmatrix} R_{11} & 0 \\ 0 & 0 \end{pmatrix} V^T \qquad \Leftrightarrow \qquad A^\dagger = V \begin{pmatrix} R_{11}^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^T.$$

The following results relate the generalized inverse, the pseudoinverse and the least squares solution to one another.

Lemma 34. *For a rectangular* $m \times n$ *matrix* A *with rank* $r \leq n \leq m$*, there is an* $n \times (n - r)$ *matrix* B *whose columns form a basis for* $\mathrm{null}(A)$*, such that the set of generalized inverses can be written as*

$$A^- = A^\dagger + BY,$$

where Y *is an arbitrary* $(n - r) \times m$ *matrix. If* A *has full rank* $r = n$*, then there is only one generalized inverse:* $A^- = A^\dagger$.

The matrix $AA^\dagger$ is a projection matrix that plays an important role in the analysis of least squares problems. In terms of the pseudoinverse, the orthogonal projections onto the fundamental subspaces associated with A can be expressed as

$$\text{Projection onto the column space of } A\text{: } \mathcal{P}_{\mathrm{range}(A)} = AA^\dagger.$$

$$\text{Projection onto the null space of } A^T\text{: } \mathcal{P}_{\mathrm{null}(A^T)} = I - AA^\dagger.$$

Theorem 35. *If A is rank deficient with a generalized inverse A^-, then any least squares solution $\boldsymbol{x}^* = \operatorname{argmin}_{\boldsymbol{x}} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ can be expressed in terms of A^- as*

$$\boldsymbol{x}^* = A^-\boldsymbol{b} = \widehat{\boldsymbol{x}}^* + \boldsymbol{z}, \qquad \boldsymbol{z} \in \operatorname{null}(A),$$

where $\widehat{\boldsymbol{x}}^*$ is the solution of minimal l_2-norm, given by

$$\boxed{\widehat{\boldsymbol{x}}^* = A^\dagger \boldsymbol{b}.} \tag{3.1.1}$$

The norm of the corresponding least squares residuals is

$$\|\boldsymbol{r}^*\|_2 = \|\widehat{\boldsymbol{r}}^*\|_2 = \|(I - AA^\dagger)\boldsymbol{b}\|_2 = \|P_{\operatorname{null}(A^T)}\boldsymbol{b}\|_2.$$

In other words:

- If A is a full-rank matrix, the LSQ problem has a unique solution that can be expressed using the pseudoinverse: $\boldsymbol{x}^* = A^\dagger \boldsymbol{b}$.
- If A is rank deficient, there are infinitely many solutions to the LSQ problem, and they can be expressed by generalized inverses; the minimum-norm solution is $\widehat{\boldsymbol{x}}^* = A^\dagger \boldsymbol{b}$.

We emphasize that the pseudoinverse is mainly a theoretical tool for analysis of the properties of least squares problems. For numerical computations one should use methods based on orthogonal transformations and factorizations, as discussed in the previous sections. We give a more detailed discussion of these methods in the next chapters.

An important result, needed later for the sensitivity analysis of the least squares solution, is the continuity of the pseudoinverse. In fact, the pseudoinverse of $A+\Delta A$, with ΔA a perturbation matrix, is a well-behaved function as long as the rank of the matrix does not change.

Theorem 36. *If* $\operatorname{rank}(A + \Delta A) = \operatorname{rank}(A)$ *and* $\eta = \|A^\dagger\|_2 \, \|\Delta A\|_2 < 1$, *then the pseudoinverse of the perturbed matrix is bounded:*

$$\|(A + \Delta A)^\dagger\|_2 \leq \frac{1}{1-\eta} \|A^\dagger\|_2.$$

Note that rounding errors always introduce perturbations of A. In terms of data fitting we can think of ΔA as also representing inaccuracies in the model $M(\boldsymbol{x}, t)$; the above theorem gives an upper bound for the size of these errors that ensures continuity of the pseudoinverse and *a fortiori* that of the LSQ solution. If the rank of A changes due to the perturbation ΔA, the pseudoinverse of the perturbed matrix may become unbounded as shown, e.g., in [20], p. 26.

In a later chapter we will also need the following formula from [101] for the derivative of an orthogonal projection.

Lemma 37. *Let $A = A(\alpha) \in \mathbb{R}^{m\times n}$ be a matrix of local constant rank, differentiable with respect to the vector of variables $\alpha \in \mathbb{R}^p$, and let $A^\dagger$ be the pseudoinverse of A. Then*

$$\frac{d}{d\alpha}(AA^\dagger) = \mathcal{P}_{\mathrm{null}(A^T)}\frac{dA}{d\alpha}A^\dagger + (A^\dagger)^T\frac{dA^T}{d\alpha}\mathcal{P}_{\mathrm{null}(A^T)},$$

where the notation $\frac{dA}{d\alpha}$ stands for taking the gradient of each component of A and constructing a three-dimensional tensor of partial derivatives (this is also called the Fréchet derivative).

3.2 The singular value decomposition

The singular value decomposition (SVD) is a very useful tool, both theoretically and computationally. Given the matrix A, the SVD provides sets of orthogonal basis for $\mathbb{R}^m$ and $\mathbb{R}^n$ such that the mapping associated with A is represented by a diagonal matrix.

Theorem 38. ***SVD.*** *Let A be a real $m \times n$ matrix with rank r satisfying $r \le n \le m$. Then there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ and a diagonal matrix $\Sigma \in \mathbb{R}^{m\times n}$ such that*

$$\boxed{A = U\,\Sigma\,V^T, \qquad \Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & 0 & \end{pmatrix}.} \tag{3.2.1}$$

The diagonal elements σ_i of Σ are the singular values of A, and they appear in descending order. The rank r is determined by the number of positive singular values; i.e., $rank(A) = r$ iff $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_n = 0$.

The columns of $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m)$ and $V = (\boldsymbol{v}_1, ..., \boldsymbol{v}_n)$ are the left and right singular vectors associated with σ_i; they form an orthonormal basis for $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively.

A simpler form of the decomposition omits the zero singular values, giving rise to the so-called economical (or thin) version of the SVD.

Theorem 39. ***Economical SVD.*** *Let A be an $m \times n$ matrix of rank r satisfying $r \le n \le m$. Then A can be factorized as $A = U_r\,\Sigma_r\,V_r^T$, where Σ_r is an $r \times r$ diagonal matrix with positive elements. Moreover, the $m \times r$ matrix U_r satisfies $U_r^T U_r = I_r$ and the $n \times r$ matrix V_r satisfies $V_r^T V_r = I_r$.*

The next theorem expresses a rank-r matrix as a sum of rank-one matrices.

Theorem 40. *If the matrix A has rank r and singular value decomposition $A = U\,\Sigma\,V^T$, with the matrices defined in Theorems 38 or 39, then A can be written as a so-called dyadic decomposition of rank-1 matrices:*

$$A = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T.$$

There are complex versions of these theorems when A has complex entries, where the matrices U and V are unitary (the singular values are always real). We summarize some useful results for working with the SVD:

- Pre- or post-multiplication by an orthogonal matrix Q does not alter the singular values of A. In particular, any permutation of the rows or columns of A does not alter its singular values.
- $A\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$ and $A^T\boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i$ for $i = 1, \ldots, n$, while $A^T\boldsymbol{u}_i = 0$ for $i = n+1, \ldots, m$.
- In addition, if $r < n$, then $A\boldsymbol{v}_i = 0$ and $A^T\boldsymbol{u}_i = 0$ for $i = r+1, \ldots, n$.
- The sets of left singular vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r\}$ and $\{\boldsymbol{u}_{r+1}, \ldots, \boldsymbol{u}_m\}$ are, respectively, orthonormal bases for range(A) and its orthogonal complement null(A^T).
- The sets of right singular vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r\}$ and $\{\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_n\}$ are, respectively, orthonormal bases for range(A^T) and its orthogonal complement null(A).
- Spectral norm: $\|A\|_2 = \sigma_1$.
- Frobenius norm: $\|A\|_{\mathrm{F}} = \left(\sigma_1^2 + \sigma_2^2 + \ldots + \sigma_r^2\right)^{1/2}$.
- The SVD is related to the eigenproblem of the symmetric, non-negative definite matrices A^TA and AA^T. In fact, the eigenvalue-vector decompositions of these matrices are

$$V^T(A^TA)\,V = \Sigma^2 = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_r^2, \overbrace{0, \ldots, 0}^{n-r}),$$

$$U^T(AA^T)\,U = \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_r^2, \underbrace{0, \ldots, 0}_{m-r}).$$

- The unique σ_i^2 correspond to the eigenvalues of A^TA (but calculating them this way is not a stable numerical process and is therefore not recommended).

- The $\boldsymbol{v}_i$ are the unique eigenvectors for simple σ_i^2. For multiple eigenvalues there is a unique associated subspace, but the orthogonal basis vectors that span the subspace and constitute the corresponding right singular vectors are not unique.
- The $\boldsymbol{u}_i$ for $i = 1, ..., n$ can, in principle, be computed from $A\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$. The $\{\boldsymbol{u}_i\}$-set is then completed up to m columns with arbitrary orthogonal vectors. Hence, a matrix may have many singular value decompositions.

The singular values of a matrix are well conditioned with respect to changes in the elements of the matrix. The following theorem is directly related to a theorem for an associated eigenvalue problem (see [150] section 5 or [20] p. 14).

Theorem 41. *Let A and ΔA be $m \times n$ matrices and denote the singular values of A and $A + \Delta A$ by σ_i and $\widetilde{\sigma}_i$, respectively. Then*

$$|\sigma_i - \widetilde{\sigma}_i| \leq \|\Delta A\|_2, \quad i = 1, \ldots, n \quad \textit{and} \quad \sum |\sigma_i - \widetilde{\sigma}_i|^2 \leq \|\Delta A\|_{\mathrm{F}}^2.$$

In other words, the perturbation in the singular values is at most as large as the perturbation in the matrix elements. We cite from [20] p. 14 the following perturbation theorem for singular vectors.

Theorem 42. *Let the SVD of A be as in Theorem 38 and let the singular values and vectors of the perturbed matrix $A + \Delta A$ be $\widetilde{\sigma}_i$, $\widetilde{\boldsymbol{u}}_i$ and $\widetilde{\boldsymbol{v}}_i$. Let $\theta(\boldsymbol{u}_i, \widetilde{\boldsymbol{u}}_i)$ and $\theta(\boldsymbol{v}_i, \widetilde{\boldsymbol{v}}_i)$ denote the angles between the exact and perturbed singular vectors and let $\gamma_i = \min(\sigma_{i-1} - \sigma_i\,,\, \sigma_i - \sigma_{i+1})$. Then*

$$\max\left\{\sin\theta(\boldsymbol{u}_i, \widetilde{\boldsymbol{u}}_i)\,,\, \sin\theta(\boldsymbol{v}_i, \widetilde{\boldsymbol{v}}_i)\right\} \leq \frac{\|\Delta A\|_2}{\gamma_i - \|\Delta A\|_2},$$

provided that $\|\Delta A\|_2 < \gamma_i$.

Thus, if σ_i is well separated from its neighbors, its corresponding singular vectors $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are well defined with respect to perturbations. On the other hand, if σ_i is close to a neighboring singular value, then their corresponding singular vectors are possibly ill-defined. For a cluster of singular values, the individual singular vectors may be inaccurate, but if the cluster is well separated from the other singular values, then the subspace itself, defined by the span of the singular vectors, is well defined (see [105], p. 450). We note in passing that for the particular case of bidiagonal matrices, the bounds depend on the relative gap between a singular value and its neighbors, and therefore the errors produced by a perturbation are much smaller [60].

The SVD can be used to show how close a given matrix is to the set of matrices with lower rank and, in particular, how close it is to being

rank deficient. This is frequently used to determine numerical rank, as the following Eckart-Young-Mirski theorem [72] states.

Theorem 43. *Let A be an $m \times n$ matrix of rank r and define the rank-k matrix*

$$A_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T.$$

Then A_k is the rank-k matrix closest to A and the distance is

$$\|A - A_k\|_2 = \sigma_{k+1}, \qquad \|A - A_k\|_{\mathrm{F}} = \sigma_{k+1}^2 + \cdots + \sigma_r^2.$$

Given the singular value decomposition of a matrix A, we have the necessary tool to give an explicit expression for the pseudoinverse in terms of the SVD components. Specifically, the pseudoinverse of a general real, rectangular matrix A of rank r is given by

$$A^\dagger = V \Sigma^\dagger U^T, \ \Sigma = (\, \Sigma_1^\dagger \,,\, 0\,), \ \Sigma = \mathrm{diag}(\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0).$$

The Moore-Penrose conditions are easily verified for this matrix. Notice that

$$\|A^\dagger\|_2 = \sigma_r^{-1}$$

for a rank-r matrix A.

Theorem 44. *The minimum-norm LSQ solution (that is identical to the LSQ solution $\boldsymbol{x}^*$ when A has full rank) is given by*

$$\boxed{\widehat{\boldsymbol{x}}^* = A^\dagger \boldsymbol{b} = V \Sigma^\dagger U^T \boldsymbol{b} = \sum_{i=1}^{r} \frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} \boldsymbol{v}_i,} \tag{3.2.2}$$

and the corresponding solution and residual norms are

$$\|\widehat{\boldsymbol{x}}^*\|_2^2 = \sum_{i=1}^{r} \left(\frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} \right)^2, \qquad \|\boldsymbol{r}^*\|_2^2 = \sum_{i=r+1}^{m} (\boldsymbol{u}_i^T \boldsymbol{b})^2. \tag{3.2.3}$$

All these results can be extended to the complex case.

Example 45. *We have already seen the normal equations and the QR factorization for the NMR problem from Example 1; here we show the SVD analysis of this problem:*

$$\Sigma_3 = \begin{pmatrix} 7.46 & 0 & 0 \\ 0 & 2.23 & 0 \\ 0 & 0 & 0.59 \end{pmatrix},$$

$$V_3 = \begin{pmatrix} 0.114 & 0.613 & 0.782 \\ 0.312 & 0.725 & -0.614 \\ 0.943 & -0.314 & 0.109 \end{pmatrix}, \qquad U_3^T \boldsymbol{b} = \begin{pmatrix} 7.846 \\ 4.758 \\ -0.095 \end{pmatrix},$$

and therefore

$$\frac{\boldsymbol{u}_1^T \boldsymbol{b}}{\sigma_1} = 1.052, \qquad \frac{\boldsymbol{u}_2^T \boldsymbol{b}}{\sigma_2} = 2.134, \qquad \frac{\boldsymbol{u}_3^T \boldsymbol{b}}{\sigma_3} = -0.161.$$

Hence, the LSQ solution is expressed in terms of the SVD as

$$\begin{aligned} \boldsymbol{x}^* &= \sum_{i=1}^{3} \frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} \boldsymbol{v}_i \\ &= 1.052 \begin{pmatrix} 0.114 \\ 0.312 \\ 0.943 \end{pmatrix} + 2.134 \begin{pmatrix} 0.613 \\ 0.725 \\ -0.314 \end{pmatrix} - 0.161 \begin{pmatrix} 0.782 \\ -0.614 \\ 0.109 \end{pmatrix} \\ &= \begin{pmatrix} 1.303 \\ 1.973 \\ 0.305 \end{pmatrix}. \end{aligned}$$

3.3 Generalized singular value decomposition

For some of the material on least squares problems with constraints, a generalized version of the SVD is needed. Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times n}$, the generalized SVD (GSVD) provides sets of orthogonal bases in $\mathbb{R}^m$, $\mathbb{R}^n$ and $\mathbb{R}^p$, such that the mappings associated with both A and B are represented by diagonal matrices.

Theorem 46. *Given $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times n}$, there are orthogonal matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{p \times p}$ and a nonsingular matrix $X \in \mathbb{R}^{n \times n}$ such that $U^T A\, X = D_A = \operatorname{diag}(\alpha_1, \ldots, \alpha_n)$, $V B\, X = D_B = \operatorname{diag}(\beta_1, \ldots, \beta_q)$, where $\alpha_i \geq 0$ for $i = 1, \ldots, n$, $\beta_i \geq 0$ for $i = 1, \ldots, q$ and $q = \min(n, p)$.*

The generalized SVD is related to the generalized eigenvalue problem: there is a nonsingular matrix X that is common to both matrices, such that both $X^T(A^T A)\, X$ and $X^T(B^T B)\, X$ are diagonal.

The definition of the GSVD in [20] is slightly stronger; it assumes that the diagonal elements are ordered and involves the rank of $\begin{pmatrix} A \\ B \end{pmatrix}$. The generalized SVD is but one of several generalized orthogonal decompositions that are designed to avoid the explicit computation, for accuracy reasons, of some matrix operations. In the present case, if A and B are square and B is nonsingular, then the GSVD corresponds to the SVD of AB^{-1}.

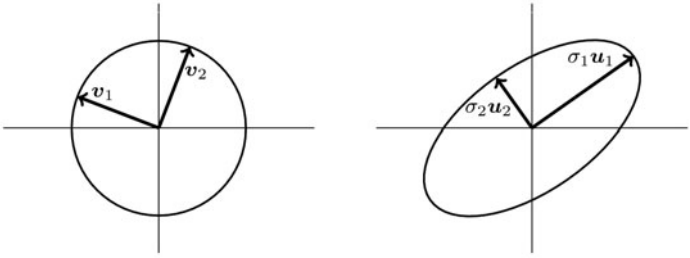

Figure 3.4.1: The geometric interpretation of the SVD, based on the relation $A\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$ for $i = 1, \ldots, n$. The image $A\mathcal{S}$ of the hyper-sphere $\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x}\|_2 = 1\}$ is a hyper-ellipsoid in $\mathbb{R}^m$ centered at the origin, with principal axes being the singular vectors $\boldsymbol{u}_i$ and with the singular values σ_i as the half-axes lengths. The condition number of A is the eccentriticy, i.e., the ratio between the largest and smallest half-axes lengths.

3.4 Condition number and column scaling

The definition of the condition number given for square nonsingular matrices can now be extended in a natural way to rectangular matrices, including those that are rank deficient.

Definition 47. *The l_2 (or spectral) condition number of a rectangular rank-r matrix with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0$ is defined by*

$$\boxed{l_2 \text{ condition number: } \operatorname{cond}(A) = \frac{\sigma_1}{\sigma_r}.} \tag{3.4.1}$$

If the columns of A were orthonormal, then the condition number would be $\operatorname{cond}(A) = 1$, indicating a perfectly conditioned matrix. As we shall see below, the condition number plays an important role in the study of the sensitivity of the LSQ problem to data, model and computational errors.

One can visualize the relation between the singular values and the condition number of A by considering the image by A of the unit hyper-sphere $\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x}\|_2 = 1\}$ with center at the origin. The image $A\mathcal{S}$ is a hyper-ellipsoid in $\mathbb{R}^m$ centered at the origin, with the singular vectors $\boldsymbol{u}_i$ as principal axes and with the singular values σ_i as the half-axes lengths.

This is easily deduced from the facts that the $\{\boldsymbol{v}_i\}_{i=1,\ldots,n}$ are an orthonormal basis for $\mathbb{R}^n$, $\boldsymbol{v}_i \in \mathcal{S}$ and $A\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$. See also Figure 3.4.1. The eccentricity of the hyper-ellipsoid, i.e., the ratio between the largest and the smallest half-axes, is equal to $\operatorname{cond}(A)$. Note that if A is rank deficient then the hyper-ellipsoid has collapsed to a smaller dimension.

The elongation of the hyper-ellipsoid gives a good graphical idea of the condition number of the matrix and its nearness to singularity. For a

perfectly conditioned matrix the hyper-ellipsoid becomes a hyper-sphere. Full-rank matrices generate a hyper-ellipsoid whose eccentricity increases with the condition number, and in the rank-deficient case the presence of zero singular values implies that the hyper-ellipsoid has collapsed in the $n - r$ directions $\boldsymbol{u}_i$ corresponding to $\sigma_i = 0$.

One can pre-process the original LSQ problem by scaling to improve the condition. The two options are row and column scaling. Row scaling involves left multiplication of both A and $\boldsymbol{b}$ by a diagonal matrix; however, this is equivalent to the use of weights and therefore changes the LSQ problem and its solution. Column scaling, on the other hand, involves a right multiplication of A by a diagonal matrix D corresponding to a scaling of the variables.

Column scaling is therefore a candidate technique to improve the condition number of the matrix in the LSQ problem, adding robustness to the numerical least squares computations. The LSQ problem changes to

$$\min_{\boldsymbol{x}} \|\boldsymbol{b} - ADD^{-1}\boldsymbol{x}\|_2 = \min_{\boldsymbol{y}} \|\boldsymbol{b} - AD\boldsymbol{y}\|_2 \qquad \text{with} \qquad \boldsymbol{y} = D^{-1}\boldsymbol{x}.$$

If A has full rank, the unique solution $\boldsymbol{x}^*$ is obtained correctly from $\boldsymbol{y}^*$. However, if A is rank deficient, although the correct set of residual-minimizing vectors is obtained, the minimal-length vector $\widehat{\boldsymbol{y}}^*$ is chosen to minimize $\left\|D^{-1}\boldsymbol{x}\right\|_2$, not $\|\mathbf{x}\|_2$, and therefore the computed solution $\boldsymbol{x} = D\widehat{\boldsymbol{y}}^*$ does not correspond to the correct minimum-norm solution $\widehat{\boldsymbol{x}}^*$.

The following theorem from [243] advises on an appropriate choice of the diagonal scaling matrix D in order to reduce as much as possible the condition number $\mathrm{cond}(AD)$ of the column-scaled matrix.

Theorem 48. *Let B denote a symmetric positive definite matrix whose diagonal elements are identical. Then*

$$\mathrm{cond}(B) \leq n \min_{D} \mathrm{cond}(D^T B\, D),$$

where D varies over the set of all diagonal matrices.

This theorem, when applied to the normal equations matrix $A^T A$, states that, in the full-rank case, if the scaling D is such that all columns of AD have unit Euclidean norm – so that the diagonal elements of $A^T A$ are 1 – then the condition number of the matrix AD is within a factor $\sqrt{n}$ of the optimum $\min_D \mathrm{cond}(AD)$. Hence, unless there is additional statistical knowledge, the model basis functions $f_j(t)$ in (1.2.3) should preferably be scaled, so that all columns of A are of unit Euclidean length. The effect can be important, as shown by the following example from [20], p. 50.

Example 49. *In data fitting with polynomials one often uses the monomials $f_j(t) = t^{j-1}$ for $j = 1, \ldots, n$ as the model basis functions. Then the*

matrix is a Vandermonde matrix with elements $a_{ij} = t_i^{j-1}$, *where* $t_1, \ldots, t_m$ *are the data abscissas. In this example we use* $m = 21$, $n = 6$ *and the abscissas* $0, 1, 2, \ldots, m$. *The corresponding Vandermonde matrix has elements* $(i-1)^{j-1}$ *(there is a misprint in [20]) and condition number* $\text{cond}(A) = 6.40 \cdot 10^6$. *Scaling its columns to unit Euclidean norm reduces the condition number to* $\text{cond}(AD) = 2.22 \cdot 10^3$ – *a dramatic reduction.*

It is interesting to know that in the square case there are fast algorithms (requiring $O(n^2)$ flops) for solving Vandermonde systems that take advantage of the special structure and are fairly impervious to this generic ill-conditioning [27]. Higham [128] has provided a detailed analysis showing why this algorithm performs as well as it does. This alerts us to the fact that condition numbers are problem and algorithm dependent. In fact, for ill-conditioned Vandermonde systems Gaussian elimination blows up (i.e., it is unstable).

Demeure and Sharf [62] derived a fast algorithm to compute the QR factors of a complex Vandermonde matrix with complexity $O(5mn)$ flops. No discussion of the numerical stability of the method is given. See also [206].

Example 50. *We return to the NMR problem from Example 1, whose normal equation matrix* A^TA *is shown in Example 11. It follows from the SVD analysis in Example 45 that* $\text{cond}(A) = \sigma_1/\sigma_3 = 7.46/0.59 = 12.64$. *The optimal column scaling for this matrix is approximately*

$$D_{\text{opt}} = \text{diag}(0.659, 0.438, 0.141),$$

leading to the condition number $\text{cond}(AD_{\text{opt}}) = 6.09$ *(there is an arbitrary scaling in* D_{opt} *which we resolved by setting* $(d_{\text{opt}})_{33} = \|A(:,3)\|_2^{-1}$*). The scaling* D *that normalizes the columns of* A *is*

$$D = \text{diag}(0.697, 0.350, 0.141),$$

leading to $\text{cond}(AD) = 6.21$, *which is just slightly larger than the optimal one.*

One important last observation is that column scaling affects the singular values; in fact, scaling may lead to a better computation of the numerical rank (see the next section). This is particularly important in the case of nearly rank-deficient matrices. For a careful discussion of scaling, especially the use of statistical information to define scaling matrices, see [150], chapter 25 and [104].

Example 51. *We finish this section with another example related to polynomial fitting with monomials as basis functions. As we have already seen, the condition number of the associated Vandermonde matrix can be quite*

n	cond(A)	cond(A')	cond(A'')
3	$5.18 \cdot 10^2$	21.0	3.51
4	$1.16 \cdot 10^4$	$1.10 \cdot 10^2$	7.51
5	$2.65 \cdot 10^5$	$5.93 \cdot 10^2$	17, 2
6	$6.40 \cdot 10^6$	$3.26 \cdot 10^3$	38.9
7	$1.66 \cdot 10^8$	$1.8e \cdot 10^4$	91.9
8	$4.58 \cdot 10^9$	$1.05 \cdot 10^5$	$2.16 \cdot 10^2$
9	$1.33 \cdot 10^{11}$	$6.17 \cdot 10^5$	$5.29 \cdot 10^2$
10	$4.06 \cdot 10^{12}$	$3.73 \cdot 10^6$	$1.29 \cdot 10^3$

Table 3.1: Condition numbers for Vandermonde matrices of size $21 \times n$ in polynomial data fitting with monomials as basis functions. The matrix A corresponds to using the abscissas $0, 1, 2, \ldots, 20$, while A' and A'' arise when the t-interval is transformed to $[0, 1]$ and $[-1, 1]$, respectively.

large because of the growth of the polynomials. Hence, a way to improve the conditioning of the matrix is to apply a change of variables in order to reduce the t-interval. Table 3.1 shows the condition numbers for the $21 \times n$ *matrices associated with the three invervals* $[0, 21]$*,* $[0, 1]$ *and* $[-1, 1]$ *for fitting orders* $n = 3, \ldots, 10$*; clearly, the interval* $[-1, 1]$ *gives the best-conditioned problems.*

3.5 Perturbation analysis

We finish this chapter with a study of the robustness of the least squares solution under perturbations. We distinguish between two different kinds of perturbations, namely, data errors in the form of measurement errors, which manifest themselves as a perturbation of the right-hand side, and model errors in the form of errors in the fitting model $M(\boldsymbol{x}, t)$, which manifest themselves as perturbations of the matrix. At this stage we ignore the rounding errors during the computations; we return to this aspect in the next chapter.

When assessing the quality of a least squares solution and depending on the problem, one might be more interested in a bound for either the LSQ solution $\boldsymbol{x}^*$ (for parameter estimation problems) or for the residual $\boldsymbol{r}^*$ (for data approximation problems).

As is well known from the analysis of linear systems of equations, cond(A) plays an important role in sensitivity studies. In particular, we know that if the matrix is ill-conditioned, the solution of the linear system may be very sensitive to perturbations. We shall see that cond(A) is also important for the least squares problem, but we will also show that it does not tell the full story about the sensitivity of the least squares solution.

This will motivate the introduction of the condition number $\mathrm{cond}(A, \boldsymbol{b})$ for the least squares problem.

Throughout this section we use the following standard notation for perturbation analysis. The unperturbed problem $\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2$ has the LSQ solution $\boldsymbol{x}^*$ and the corresponding residual $\boldsymbol{r}^*$. The perturbed system has matrix $A + \Delta A$ and right-hand side $\boldsymbol{b} + \Delta \boldsymbol{b}$; we write the perturbed solution as $\boldsymbol{x}^* + \Delta \boldsymbol{x}$ and the corresponding solution as $\boldsymbol{r}^* + \Delta \boldsymbol{r}$. We are interested in deriving upper bounds for the norms of the perturbations $\Delta \boldsymbol{x}$ and $\Delta \boldsymbol{r}$.

We have seen that the least squares solution $\boldsymbol{x}^*$ is the solution to the normal equations (2.1.5) – when A has full rank – and one can therefore, in principle, use the perturbation bounds available for linear systems of equations.

Example 52. ***Sensitivity analysis via normal equations.*** *It follows immediately from the normal equations that if* $\Delta A = 0$ *then*

$$\|\Delta \boldsymbol{x}\|_2 = \|(A^T A)^{-1} A^T \Delta \boldsymbol{b}\|_2 \leq \|(A^T A)^{-1}\|_2 \, \|A\|_2 \, \|\Delta \boldsymbol{b}\|_2.$$

Moreover, we have $\|A\,\boldsymbol{x}^*\|_2 \leq \|A\|_2 \, \|\boldsymbol{x}^*\|_2$. *From the SVD of* A *we know that* $\|A\|_2 = \sigma_1$ *and* $\|(A^T A)^{-1}\|_2 = \max\{\sigma_i^{-2}\} = \sigma_n^{-2}$. *Combining these bounds and using* $\mathrm{cond}(A) = \sigma_1/\sigma_n$, *we obtain the following perturbation bound for the solution:*

$$\frac{\|\Delta \boldsymbol{x}\|_2}{\|\boldsymbol{x}^*\|_2} \leq \mathrm{cond}(A)^2 \, \frac{\|\Delta \boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2}.$$

The upper bound in this expression is too pessimistic, due to the presence of the squared condition number; below we give attainable perturbation bounds, and we show that a squared condition number only arises when matrix perturbations are present. However, we emphasize that the rounding errors in the normal equations approach are proportional to $\mathrm{cond}(A)^2$.

Even as a measure of the sensitivity of the solution of a linear system, the matrix condition number may be too conservative. Chan and Foulser [41] demonstrate how the projection of the right-hand side $\boldsymbol{b}$ onto the range of A, in addition to $\mathrm{cond}(A)$, can strongly affect the sensitivity of the linear system solution in some instances. Thus, the definition of the condition number of a least squares problem that takes into account the right-hand side is a natural step.

The following theorem, adapted from [20], p. 31, applies to the full-rank case. It gives *attainable* upper bounds for the norms of the solution and residual perturbations in terms of perturbations of A and $\boldsymbol{b}$. These bounds are obtained under the hypothesis that the components of A and $\boldsymbol{b}$ have all errors of roughly the same order. Note that the use of norms ignores how the perturbations are actually distributed among the components of an array. For component-wise bounds see [21].

Theorem 53. ***LSQ perturbation bound.*** *Assume that A has full rank and that $\boldsymbol{x}^*$ minimizes $\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2$ with residual $\boldsymbol{r}^*$. Let ΔA, $\Delta\boldsymbol{b}$ and $\Delta\boldsymbol{x}$ be perturbations to A, $\boldsymbol{b}$ and $\boldsymbol{x}^*$, so that $\boldsymbol{x}^* + \Delta\boldsymbol{x}$ minimizes $\|(\boldsymbol{b} + \Delta\boldsymbol{b}) - (A + \Delta A)(\boldsymbol{x} + \Delta\boldsymbol{x}\|_2$ with residual $\boldsymbol{r}^* + \Delta\boldsymbol{r}$. Moreover, assume that*

$$\frac{\|\Delta A\|_2}{\|A\|_2} \leq \epsilon_A, \qquad \frac{\|\Delta\boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2} \leq \epsilon_{\boldsymbol{b}}$$

and that $\operatorname{cond}(A)\,\epsilon_A < 1$. *Then the perturbation in the solution satisfies*

$$\boxed{\frac{\|\Delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}^*\|_2} \leq \frac{\operatorname{cond}(A)}{1 - \operatorname{cond}(A)\,\epsilon_A}\left(\epsilon_A + \frac{\epsilon_b\|\boldsymbol{b}\|_2}{\|A\|_2\,\|\boldsymbol{x}^*\|_2} + \frac{\operatorname{cond}(A)\,\epsilon_A\|\boldsymbol{r}^*\|_2}{\|A\|_2\,\|\boldsymbol{x}^*\|_2}\right)} \tag{3.5.1}$$

and in the residual

$$\boxed{\frac{\|\Delta\boldsymbol{r}\|_2}{\|\boldsymbol{r}^*\|_2} \leq \epsilon_A\left(\operatorname{cond}(A) + \frac{\|A\|_2\,\|\boldsymbol{x}^*\|_2}{\|\boldsymbol{r}^*\|_2}\right) + \epsilon_{\boldsymbol{b}}\,\frac{\|\boldsymbol{b}\|_2}{\|\boldsymbol{r}^*\|_2}.} \tag{3.5.2}$$

The condition $\operatorname{cond}(A)\,\epsilon_A < 1$ *ensures that the rank of A does not change under the perturbation.*

We emphasize that when matrix perturbations are present then the solution's sensitivity to these errors is proportional to $\operatorname{cond}(A)^2$, while the sensitivity of the residual is proportional to $\operatorname{cond}(A)$.

If we only consider data errors, i.e., if $\epsilon_A = 0$, then the perturbation bounds for the LSQ solution and the residual take the simpler forms

$$\frac{\|\Delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}^*\|_2} \leq \operatorname{cond}(A)\,\epsilon_{\boldsymbol{b}}\,\frac{\|\boldsymbol{b}\|_2}{\|A\|_2\,\|\boldsymbol{x}^*\|_2}, \qquad \frac{\|\Delta\boldsymbol{r}\|_2}{\|\boldsymbol{r}^*\|_2} \leq \epsilon_{\boldsymbol{b}}\,\frac{\|\boldsymbol{b}\|_2}{\|\boldsymbol{r}^*\|_2}, \tag{3.5.3}$$

which are similar to the bounds for linear systems of equations where $\boldsymbol{b} = A\,\boldsymbol{x}$. We see that the condition number $\operatorname{cond}(A)$ governs the data perturbations in this case.

Let us now consider the opposite case, with only model errors, i.e., $\epsilon_{\boldsymbol{b}} = \boldsymbol{0}$. Then the perturbation bound reduces to

$$\frac{\|\Delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}^*\|_2} \leq \frac{\operatorname{cond}(A)}{1 - \operatorname{cond}(A)\,\epsilon_A}\left(1 + \operatorname{cond}(A)\,\frac{\|\boldsymbol{r}^*\|_2}{\|A\|_2\,\|\boldsymbol{x}^*\|_2}\right)\epsilon_A,$$

and we recall that the last factor ϵ_A is a bound for the relative perturbation in A. Inspired by this result it is common to define the condition number of the LSQ problem with respect to model perturbations as

$$\operatorname{cond}(A, \boldsymbol{b}) = \left(1 + \operatorname{cond}(A)\,\frac{\|\boldsymbol{r}^*\|_2}{\|A\|_2\,\|\boldsymbol{x}^*\|_2}\right)\operatorname{cond}(A). \tag{3.5.4}$$

We note that the least squares problem condition number, $\mathrm{cond}(A, \boldsymbol{b})$, includes a term with the squared matrix condition number, although multiplied by the relative residual norm. The other important difference from the traditional matrix condition number is its explicit dependence on $\boldsymbol{b}$, through $\boldsymbol{x}^*$ and $\boldsymbol{r}^*$. If A is ill-conditioned, then both condition numbers are large. One can bound $\mathrm{cond}(A, \boldsymbol{b})$ from below by $\sec(\theta)\,\mathrm{cond}(A) \leq \mathrm{cond}(A, b)$, cf. [108]. Here, θ is the angle between the right-hand side $\boldsymbol{b}$ and the subspace $\mathrm{range}(A)$, i.e., a measure of the consistency of the problem. From this lower bound it is straightforward to see that the problem is ill-conditioned if either the matrix is ill-conditioned (i.e., $\mathrm{cond}(A)$ is large) or $\boldsymbol{b}$ is almost perpendicular to $\mathrm{range}(A)$ (i.e., $\sec(\theta)$ is large). The following example based on an example from [105] with a modification by Grcar [108] illustrates this aspect

Example 54. *Consider the LSQ problem*

$$A = \begin{pmatrix} 1 & 0 \\ 0 & \alpha \\ 0 & 0 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} \beta\, c \\ \beta\, s \\ 1 \end{pmatrix}, \qquad c^2 + s^2 = 1,$$

where $\alpha < 1$ *is such that* $\mathrm{cond}(A) = \alpha^{-1}$. *Moreover* β *controls the size of the component of* $\boldsymbol{b}$ *in the range of* A *and* s *controls the components of* $\boldsymbol{b}$ *along the left singular vectors* $\boldsymbol{u}_1 = (1\ 0\ 0)^T$ *and* $\boldsymbol{u}_2 = (0\ 1\ 0)^T$. *The LSQ solution, its residual and their norms are*

$$\boldsymbol{x}^* = \begin{pmatrix} \beta\, c \\ \frac{\beta}{\alpha}\, s \end{pmatrix}, \quad \boldsymbol{r} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \|\boldsymbol{x}^*\|_2 = \beta \sqrt{c^2 + \frac{s^2}{\alpha^2}}, \quad \|\boldsymbol{r}^*\|_2 = 1.$$

Thus, the problem condition number is

$$\mathrm{cond}(A, \boldsymbol{b}) = \frac{1}{\alpha} \left(1 + \frac{1}{\alpha} \frac{(c^2 + \frac{s^2}{\alpha^2})^{-1/2}}{\beta} \right).$$

There are two circumstances in which this condition number can become large. One is when α *is small, which corresponds to an ill-conditioned matrix (in which case both* $\mathrm{cond}(A)$ *and* $\mathrm{cond}(A, \boldsymbol{b})$ *become large), while the other situation is when* β *is small, which happens when the right-hand side* $\boldsymbol{b}$ *has only a small component in the range of* A *(such that the LSQ solution norm is small).*

Let us now consider how the problem condition number depends on the parameter s *that controls the components of* $\boldsymbol{b}$ *along the two left singular vectors. We have*

$$\mathrm{cond}(A, \boldsymbol{b}) \to \begin{cases} \left(1 + \frac{1}{\beta}\right) \frac{1}{\alpha}, & \text{for} \quad s \to 1 \ (c \to 0) \\ \left(1 + \frac{1}{\alpha} \frac{1}{\beta}\right) \frac{1}{\alpha}, & \text{for} \quad s \to 0 \ (c \to 1). \end{cases}$$

$s \to 1$ means that $\boldsymbol{b}$ becomes dominated by the singular vector $\boldsymbol{u}_2$ and we see that the condition number is essentially proportional to β^{-1} and α^{-1}. However, when $s \to 0$, meaning that $\boldsymbol{b}$ becomes dominated by the principal singular vector $\boldsymbol{u}_1$, then the condition number is essentially proportional to β^{-1} and α^{-2}; note the squaring of the matrix condition number $\mathrm{cond}(A)$ *in the latter case.*

The above example illustrates that the problem can be ill-conditioned even if the matrix is well conditioned; this happens when the right-hand side has only a small component in the range of A. This situation should, of course, be avoided in data fitting because it means that the model is not capable of describing the data very well. The example also illustrates under which circumstances the influence of $\mathrm{cond}(A)^2$ is noticeable, namely, when the right-hand side is dominated by components along the singular vectors corresponding to the larger singular values (which is typical for discretizations of ill-posed problems). In all cases, however, we should recall that these effects depend on the size of the LSQ residual! If the problem is almost consistent – because the data errors and approximation errors are small – then the factor $\|\boldsymbol{r}^*\|_2$ in $\mathrm{cond}(A, \boldsymbol{b})$ is also small and this condition number behaves like $\mathrm{cond}(A)$. Only for large-residual problems we see a major difference in the two condition numbers.

In practical applications all these error bounds are valuable only if there is an efficient way to compute the condition number. For most of the algorithms described in the next sections, at least an estimate of the matrix condition number $\mathrm{cond}(A)$ can be computed inexpensively using information generated while solving the problem.

In order to bound the actual error in a *computed* least squares solution, i.e., to determine its accuracy, one can combine the above error estimates with the backward stability bounds for the algorithm used, as described in detail in Appendix A. For each of the algorithms that will be described in the next chapters, the backward stability is also discussed.

We have examined the sensitivity of full-rank problems to model errors. In the case of a rank-deficient matrix A no such estimate can be derived, in general, because the solution may not be a continuous function of the matrix elements, as one can see from the discontinuity of the pseudoinverse, so one needs to restrict the size of the perturbations so that the rank remains constant.

We finish this section with an example that illustrates the connection between the SVD and the perturbations in the right-hand side. We also demonstrate that the SVD gives a more detailed picture of the sensitivity than the norm bounds.

Example 55. *We consider the simplified NMR problem from Example 12,*

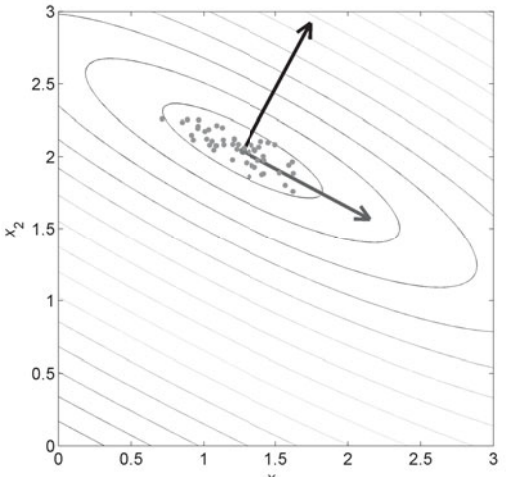

Figure 3.5.1: Level curves for simplified NMR problem.

for which the singular values and right singular vectors are

$$\sigma_1 = 3.211, \quad \sigma_2 = 0.805, \qquad \boldsymbol{v}_1 = \begin{pmatrix} 0.472 \\ 0.881 \end{pmatrix}, \quad \boldsymbol{v}_2 = \begin{pmatrix} -0.881 \\ 0.472 \end{pmatrix}.$$

Figure 3.5.1 shows the level curves for the least squares problem, where the arrows depict the right singular vectors $\boldsymbol{v}_1$ *and* $\boldsymbol{v}_2$ *respectively. The cloud of dots are the LSQ solutions for 50 perturbed problems with the same standard deviation* $\varsigma = 0.2$*. This figure illustrates that the sensitivity of the LSQ solution* $\boldsymbol{x}^* = (\boldsymbol{u}_1^T \boldsymbol{b}/\sigma_1)\, \boldsymbol{v}_1 + (\boldsymbol{u}_2^T \boldsymbol{b}/\sigma_2)\, \boldsymbol{v}_2$ *is different in the two directions given by the singular vectors* $\boldsymbol{v}_1$ *and* $\boldsymbol{v}_2$*. The solution is clearly more sensitive in the direction of the singular vector* $\boldsymbol{v}_2$ *that corresponds to the smaller singular value and less sensitive in the direction of* $\boldsymbol{v}_1$*. This is true in general – the sensitivity of the LSQ solution in the direction of* $\boldsymbol{v}_i$ *is proportional to* σ_i^{-1}*.*

This page intentionally left blank

Chapter 4

Direct Methods for Full-Rank Problems

This chapter describes a number of algorithms and techniques for the case where the matrix has full rank, and therefore there is a unique solution to the LSQ problem. We compare the efficiency of the different methods and make suggestions about their use. We start with the least expensive, albeit also the least robust method: solution of the normal equations, a convenient approach only if the LSQ problem is well conditioned and the number of rows of A is small. Next, a more robust method for slightly overdetermined problems is described. The largest part of this chapter is taken up with methods using orthogonal transformations.

A word on our notation: Throughout this chapter, $\widehat{\boldsymbol{e}}_i$ denotes the ith column of the identity matrix – not to be confused with the noise vector $\boldsymbol{e}$ that does not appear in this chapter. The computational work is measured in flops, where one "flop" is a single floating-point operation ($+$, $-$, $*$ or $/$).

4.1 Normal equations

Theorem 9 shows that the LSQ solution $\boldsymbol{x}^*$ can be obtained by solving the normal equations $A^TA\boldsymbol{x} = A^T\boldsymbol{b}$, which in the full-rank case is a symmetric positive definite system. While A is an $m \times n$ matrix, the Grammian matrix A^TA of the normal equations is $n \times n$ and thus comparatively smaller if $m \gg n$. Also, only about half of the elements (more precisely, $\frac{1}{2}n(n+1)$) of A^TA need to be computed and stored, resulting in a net data compression. The condition number of the normal equations system, however, is $\mathrm{cond}(A^TA) = \mathrm{cond}(A)^2$, thus making it potentially more sensitive to rounding errors than methods based on orthogonal transformations applied directly to A.

The natural solution method for the normal equations is the Cholesky factorization [20, 105] (known as the square root method in statistics). The total operation count for forming the triangular part of the Grammian A^TA, computing the Cholesky factor R_1, such that $A^TA = R_1^TR_1$, forming $A^T\boldsymbol{b}$, and finally solving the resulting two triangular systems, amounts to $n^2(m+\frac{1}{3}n)$ flops, which is more economical than other methods for LSQ problems. The following theorem shows that, if we work with the Grammian for the bordered matrix $\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix}$, then as a by-product we obtain the norm of the LSQ residual.

Theorem 56. *The Grammian for the bordered matrix* $\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix}$ *has the* $(n+1)\times(n+1)$ *Cholesky factor*

$$C = \begin{pmatrix} R_1 & Q_1^T\boldsymbol{b} \\ 0 & \|\boldsymbol{r}^*\|_2 \end{pmatrix},$$

where the diagonal elements of R_1 *are positive and* $A = Q_1R_1$ *is the economical QR factorization (17). Moreover,* $Q_1^T\boldsymbol{b} = R_1^{-1}(A^T\boldsymbol{b})$ *is the reduced right-hand side of the normal equations and* $\boldsymbol{r}^*$ *is the LSQ residual.*

Proof. Using the full QR factorization of A it follows immediately that

$$\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} = \begin{pmatrix} QR & QQ^T\boldsymbol{b} \end{pmatrix} = Q\begin{pmatrix} R_1 & Q_1^T\boldsymbol{b} \\ 0 & Q_2^T\boldsymbol{b} \end{pmatrix}.$$

By using an orthogonal transformation H_2 (in fact, a Householder reflection, cf. Section 4.3), we can obtain $H_2^TQ_2^T\boldsymbol{b} = (\|Q_2^T\boldsymbol{b}\|_2, 0, \ldots, 0)^T$. Since $\|Q_2^T\boldsymbol{b}\|_2 = \|Q_2Q_2^T\boldsymbol{b}\|_2 = \|\boldsymbol{r}^*\|_2$ we get

$$\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} = Q\begin{pmatrix} R_1 & Q_1^T\boldsymbol{b} \\ 0 & \|\boldsymbol{r}^*\|_2 \\ 0 & 0 \end{pmatrix},$$

showing that C is the Cholesky factor of the bordered matrix. Moreover, from $A = Q_1R_1$ we get $Q_1^T = R_1^{-1}A^T$, so that $Q_1^T\boldsymbol{b} = R_1^{-1}(A^T\boldsymbol{b})$ is the reduced right-hand side of the normal equations. □

There are some disadvantages when one uses the normal equations for ill-conditioned problems. The first one is related to the computation of the Grammian A^TA. If the columns of A are close to being linearly dependent, then there might be significant rounding error in the computation of this matrix, because there is in fact a squaring of the elements, as shown in the following example from [57].

Example 57. *Consider the nonsingular matrix*

$$A = \begin{pmatrix} 1 & 1 \\ 10^{-3} & 0 \\ 0 & 10^{-3} \end{pmatrix}.$$

If we use 6- or 7-digit chopped arithmetic, then the computed versions of the normal equations matrix are

$$f\ell_6(A^TA) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \qquad f\ell_7(A^TA) = \begin{pmatrix} 1+10^{-6} & 1 \\ 1 & 1+10^{-6} \end{pmatrix}.$$

The first matrix is singular due to the effect of the finite precision! The second matrix is computed with just enough precision to be nonsingular (but it is ill-conditioned).

The recommendation is to use extended precision accumulation for the inner products in order to diminish rounding errors and so avoid a possible breakdown of the Cholesky factorization. Wilkinson has shown that if $\varepsilon_M \text{cond}(A^TA) \leq 1$ then the Cholesky process can be completed, since no square root of a negative number arises (see [105], p. 147). This condition number can be reduced by normalizing the columns of A, so that the resulting matrix has columns of unit Euclidean norm, as discussed in the previous chapter.

The least squares solution via the normal equations is not backward stable, i.e., the computed solution is not, in general, the exact solution of a slightly different least squares problem. If $D = \text{diag}(\|A(:,j)\|_2)$, then a bound for the computed LSQ solution $\widetilde{\boldsymbol{x}}^*$ is approximately (see [128], pp. 387)

$$\frac{\left\|D(\boldsymbol{x}^* - \widetilde{\boldsymbol{x}}^*)\right\|_2}{\|D\boldsymbol{x}^*\|_2} \lesssim \mathcal{O}\left(\text{cond}(AD^{-1})^2 \varepsilon_M\right).$$

In theory, there exists a scaling D that gives a realistic upper bound for the relative errors that is as small as possible, but in practice the scaling is not so important, and experiments (illustrated by our figure) show that the numerical results are practically independent of the scaling. In short, the normal equations method is the least expensive way to solve well-conditioned problems. An example of this is data fitting with orthogonal polynomials, where A^TA is diagonal.

Example 58. *Figure 4.1.1 illustrates the accuracy of the LSQ solution when it is computed via the normal equations. We generated coefficient matrices A with condition numbers varying from* 10^2 *to* 10^{11} *and solved the LSQ problems via the normal equations, both in their standard formulation and with the columns of A scaled to unit length. We generated two types of problems – either with zero residual (i.e.,* $\boldsymbol{b} \in \text{range}(A)$*) or with a nonzero residual* $\boldsymbol{r}^*$*. The figure shows the relative errors* $\|\boldsymbol{x}^* - \widetilde{\boldsymbol{x}}^*\|_2 \,/\, \|\boldsymbol{x}^*\|_2$*, where* $\widetilde{\boldsymbol{x}}^*$ *is the computed solution, and we see that these errors are proportional to* $\text{cond}(A)^2$*. As* $\text{cond}(A)$ *approaches* 10^8 *the relative errors approach* 1*, and when* $\text{cond}(A)$ *exceeds approximately* 10^9*, then the Cholesky factorization breaks down in most cases. We note that the relative errors are, on average, the same for the standard and scaled versions of the problem.*

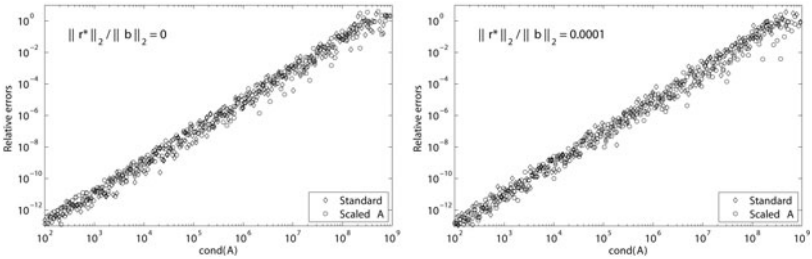

Figure 4.1.1: The relative errors of LSQ residuals computed by means of the normal equations, both in their standard form and in a variant where A is normalized to have columns of unit 2-norm. The errors for both cases are similar and proportional to $\mathrm{cond}(A)^2$. The left plot is for a LSQ problem with zero residual, while the right plot is for a problem with nonzero residual.

One practical suggestion for large problems, when secondary storage is needed for the matrix A [20], is to compute A^TA and $A^T\boldsymbol{b}$ in an *outer* product form instead of the usual inner product that requires the access to each column of A many times. The idea of the outer product form assumes that the bordered matrix $(\ A \quad \boldsymbol{b}\ )$ is stored row-wise: $[\,A(i,:)\,,\,b(i)\,]$. The matrix product A^TA can be written as the sum of rank-one matrices:

$$A^TA = \sum_{i=1}^{m} A(i,:)^T A(i,:), \qquad A^T\boldsymbol{b} = \sum_{i=1}^{m} b(i)\, A(i,:)^T.$$

For each term in the above sums, only the ith row must be retrieved, and thus it is retrieved only once.

4.2 LU factorization

A method introduced by Peters and Wilkinson [201], suited for slightly overdetermined LSQ problems with $m - n \ll n$, is to factorize the rectangular matrix A into a product of two full-rank matrices. For early uses of this factorization see [185, 186, 193] and Section 2.5 in [20] for more details. The method has been adapted successfully to sparse and weighted problems [22]. The existence of the factorization is proven in the following theorem:

Theorem 59. *If the real rectangular $m \times n$ matrix A has rank r, then it can be factorized in the form*

$$A = B\,C,$$

where both $B \in \mathbb{R}^{m\times r}$ *and* $C \in \mathbb{R}^{r\times n}$ *have full rank* r*. Although this factorization is not unique, the unique pseudoinverse is given by:*

$$A^\dagger = C^\dagger B^\dagger = C^T(C\,C^T)^{-1}(B^T B)^{-1}B^T.$$

If A is itself full rank, Gaussian elimination can be used to compute the factors of $A = L\,U$, with L a lower trapezoidal and U an upper triangular matrix. For convenience, the upper triangular matrix can be decomposed into $D\,U$, with D diagonal and U upper triangular with unit diagonal.

In order to improve stability, it is essential to use either full pivoting or the less expensive partial pivoting, in order to keep $|L_{ij}| \le 1$ (and $|U_{ij}| \le 1$ when using complete pivoting). Note that, in principle, one does not know if A is truly full rank, and as part of the elimination process one should check for pivots (i.e., the elements of D) that are small in magnitude.

Assuming that A was indeed full rank, we obtain

$$\Pi_1 A\,\Pi_2 = LDU, \qquad L = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix},$$

where Π_1 and Π_2 are permutation matrices (Π_2 is the identity if partial pivoting is used) that try to keep L well conditioned. Moreover, L_1 is $n\times n$ lower triangular and U is $n\times n$ upper triangular, both with unit diagonal. The original LSQ problem is now replaced by another LSQ problem in L:

$$\min_{\boldsymbol{y}} \|L\,\boldsymbol{y} - \Pi_1\boldsymbol{b}\|_2, \qquad \text{with} \qquad \boldsymbol{y} = DU\Pi_2^T\boldsymbol{x}. \tag{4.2.1}$$

A good option for solving the transformed problem is to use the corresponding normal equations $L^T L\,\boldsymbol{y} = L^T\Pi_1\boldsymbol{b}$. The total cost of this approach is $2n^2(m - \frac{1}{3}n^3)$ flops, and it is numerically safe because L is usually well conditioned. This is illustrated by the following example due to Noble and cited in [20].

Example 60. *Consider the matrices*

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1+\varepsilon \\ 1 & 1-\varepsilon \end{pmatrix}, \qquad A^T A = \begin{pmatrix} 3 & 3 \\ 3 & 3+2\varepsilon^2 \end{pmatrix};$$

the latter has condition number $\operatorname{cond}(A^T A) \simeq 6\,\varepsilon^{-2}$ *and is numerically singular if* ε *is less than the square root of the machine precision. On the other hand, if we use the above* LU *factorization, keeping* $|L_{ij}| \le 1$*, then*

$$L\,U = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 0 & \epsilon \end{pmatrix}, \qquad L^T L = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix},$$

with $\operatorname{cond}(L^T L) = 1.5$*, and we see that* $L^T L$ *is much better conditioned than* $A^T A$*.*

Another option due to Cline [49], for minimization of the transformed problem (4.2.1), is to reduce the matrix L by orthogonal transformations to lower triangular form,

$$\begin{pmatrix} L_1 \\ L_2 \end{pmatrix} = Q \begin{pmatrix} \overline{L} \\ 0 \end{pmatrix}, \qquad Q^T \Pi_1 \boldsymbol{b} = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{pmatrix}.$$

The resulting problem is then

$$\min_{\boldsymbol{y}} \|L\,\boldsymbol{y} - \Pi_1 \boldsymbol{b}\|_2 = \min_{\boldsymbol{y}} \left\| \begin{pmatrix} \bar{L} \\ 0 \end{pmatrix} \boldsymbol{y} - \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{pmatrix} \right\|_2,$$

i.e., one has to solve two triangular systems $\bar{L}\boldsymbol{y} = \boldsymbol{c_1}$ and $U\boldsymbol{x} = \boldsymbol{y}$. The total cost of this approach is $3n^2(m - \frac{7}{9}n)$ flops, i.e., larger than the cost of the Peters-Wilkinson algorithm. Since LU factorization is (in practice) backward stable, as well as the Householder transformations, we believe that the combined method is also backward stable. As far as we know, there is no formal analysis published.

4.3 QR factorization

The QR factorization method is based on the factorization of A into an orthogonal matrix and an upper triangular one, as described in Theorems 16 and 17, and it is a more stable – although also a more expensive – method to solve LSQ problems. The factorization can be computed in several ways, either implicitly by Householder or Givens transformations or by explicit Gram-Schmidt orthogonalization. The basic steps for the LSQ solution are

- Transformation of the original problem to a nonsingular triangular system of equations using orthogonal transformations.
- Solution by back-substitution of this system and, optionally, computation of the residual.

The actual computations, using either Householder or Givens transformation or Gram-Schmidt orthogonalization, are described below.

Householder and Givens transformations

As pointed out already, orthogonal transformations are isometric, i.e., they preserve the l_2 norm of vectors. Geometrically this means that they are either rotations or reflections. Given a vector $\boldsymbol{v}$, Householder reflections H and Givens rotations G are designed to introduce zeros in specific positions of a vector $H\boldsymbol{v}$ or $G\boldsymbol{v}$, respectively.

Householder reflections

Householder transformations have the form $H = I - \frac{2}{\boldsymbol{u}^T\boldsymbol{u}}\boldsymbol{u}\,\boldsymbol{u}^T$, where $\boldsymbol{u}$ is an arbitrary nonzero vector and thus $H\boldsymbol{v} \in \text{span}\{\boldsymbol{u}, \boldsymbol{v}\}$. Geometrically, the transformations are reflections with respect to the hyperplane with normal $\boldsymbol{u}$. Assume that we want to zero all elements of an m-vector $\boldsymbol{v}$ except the first and obtain

$$H\boldsymbol{v} = \alpha\widehat{\boldsymbol{e}}_1, \quad \text{where} \quad \alpha = \pm\,\|\boldsymbol{v}\|_2\,.$$

Then $\boldsymbol{u}$ must be chosen as $\boldsymbol{u} = \boldsymbol{v} \mp \|\boldsymbol{v}\|_2\,\widehat{\boldsymbol{e}}_1$. The reason for the sign ambiguity is to avoid cancellation in the substraction; to prevent this, the appropriate choice is $\text{sign}(\alpha) = -\text{sign}(v_1)$.

The Householder matrix need not be formed explicitly, since we just need its action on the vector $\boldsymbol{v}$:

$$H\boldsymbol{v} = \boldsymbol{v} - 2\frac{\boldsymbol{u}^T\boldsymbol{v}}{\boldsymbol{u}^T\boldsymbol{u}}\boldsymbol{u}.$$

To store the information of a Householder transformation, only one vector and one scalar are needed. The number of operations used when applying it to $\boldsymbol{v} \in \mathbb{R}^m$ is $4m$ flops. If one wants to zero all elements of an m-vector $\boldsymbol{v}$ from and including v_{k+1}, one can embed an appropriate Householder transformation $\widetilde{H}_k$ in a unit matrix, so that the transformation applied to $\boldsymbol{v}$ is

$$H_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & \widetilde{H}_k \end{pmatrix}.$$

This leaves the first $k-1$ elements of $H_k\boldsymbol{v}$ unchanged. The submatrix $\widetilde{H}_k$ is defined as before, using now the sub-vector $(v_k, v_{k+1}, \ldots, v_m)^T$ of $\boldsymbol{v}$. A block version of the Householder transformation, suited for high-performance computing, is described in [105], p. 213.

If instead of zeroing the elements in the columns of a matrix one wants to zero elements in the matrix rows, the definition of the Householder transformations will now involve the row elements, and the transformation will be applied from the right.

Givens rotations

Instead of zeroing a vector of elements at once with a Householder reflection, one can zero a single element by means of a Givens plane rotation, defined in the 2×2 case by

$$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \quad \text{with} \quad c = \cos(\vartheta), \quad s = \sin(\vartheta).$$

The product of G with a vector rotates the vector clockwise by an angle ϑ defined by c, s. Specifically, if

$$\boldsymbol{v} = \begin{pmatrix} a \\ b \end{pmatrix}, \qquad c = \frac{a}{\sqrt{a^2+b^2}}, \qquad s = \frac{b}{\sqrt{a^2+b^2}},$$

then the application of G zeros the second component of $\boldsymbol{v}$:

$$G\boldsymbol{v} = \begin{pmatrix} ca+sb \\ -sa+cb \end{pmatrix} = \begin{pmatrix} \sqrt{a^2+b^2} \\ 0 \end{pmatrix}.$$

The cost of computing c and s is 5 flops. The implementation must take into account possible under/overflow when constructing c and s (see, e.g., [20], p. 54, for a detailed algorithm). One can embed G into an $m \times m$ identity matrix in order to manipulate single elements of a vector $\boldsymbol{v} \in \mathbb{R}^m$,

$$G^{(ij)} = \begin{pmatrix} I & & & & \\ & c & & s & \\ & & I & & \\ & -s & & c & \\ & & & & I \end{pmatrix} \begin{matrix} \\ i \\ \\ j \\ \end{matrix}.$$

The matrix $G^{(ij)}$ represents a clockwise rotation of angle ϑ in the plane $\text{span}\{\widehat{\boldsymbol{e}}_i, \widehat{\boldsymbol{e}}_j\}$. Given a vector $\boldsymbol{v}$ we can choose

$$c = v_i/(v_i^2+v_j^2)^{\frac{1}{2}}, \qquad s = v_j/(v_i^2+v_j^2)^{\frac{1}{2}},$$

in order to zero the jth element v_j by multiplying $\boldsymbol{v}$ with $G^{(ij)}$. Note that there is no restriction on i, as long as $i < j$.

The rotation matrix is never explicitly formed, and the storage of $G^{(ij)}$ can actually be reduced to one scalar only, since $s = \sqrt{1-c^2}$, see [20], p. 55. As in the case of Householder transformations, one can zero elements in rows by defining appropriate Givens rotations that will now involve only row elements and are applied from the right.

The ability to selectively zero a single element makes the Givens rotation better suited than Householder transformations for sparse and structured matrices. When pre-multiplying or post-multiplying a matrix A with a Givens rotation, only the two i, j rows or columns are affected. This allows us to interchange the order of rotations and therefore to apply sets of rotations in parallel, an additional advantage.

Fast Givens rotations

To take advantage of the flexibility of the Givens transformation, several algorithms have been designed to reduce the number of operations; see, for example, [115, 116]. We will describe the method fast Givens (FG), which

eliminates the need for the square root when computing c and also halves the number of multiplications used when applying G.

Consider a 2×2 Givens rotation matrix G applied to a $2 \times n$ matrix A:

$$GA = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \end{pmatrix}.$$

The main idea behind FG is the simplification of operations when G is applied to a diagonal matrix. If $K = \mathrm{diag}(k_1, k_2)$, then

$$GK = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix} = \begin{pmatrix} ck_1 & sk_2 \\ -sk_1 & ck_2 \end{pmatrix}.$$

But this matrix can also be written as

$$GK = \begin{cases} c \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix} \begin{pmatrix} 1 & \frac{s}{c}\frac{k_2}{k_1} \\ -\frac{s}{c}\frac{k_1}{k_2} & 1 \end{pmatrix} & \text{if } c > s, \\ s \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix} \begin{pmatrix} \frac{k_2}{k_1}\frac{c}{s} & 1 \\ 1 & \frac{k_1}{k_2}\frac{c}{s} \end{pmatrix} & \text{otherwise.} \end{cases}$$

So GK is actually of the form cKC or sKS, where C and S are 2×2 matrices with only two elements different from 1. As both cases are analogous, we will work out in detail only the case when $c > s$. The FG algorithm then works with a scaling of the matrix A so that

$$GA = GKK^{-1}A \equiv GKA' = cKCA'.$$

The product CA' involves only 4 flops for each column of A'. It is straightforward to verify that no square roots are needed, not even to compute cK (for details, see [20], p. 57). If a sequence of FG transformations has to be applied, the resulting matrix GA is already in the necessary scaled form, now with the diagonal matrix cK.

There is one numerical consideration though: at each FG application, the diagonal scaling matrix is multiplied by a factor c or s, smaller in magnitude than 1, so there is a danger of underflow and some rescaling may need to be applied [3]. Again, as with the previous transformations, for applications to a general $m \times n$ matrix, the transformations are embedded into an $m \times m$ identity matrix.

Summary of Householder and Givens transformations

First, a couple of comments: only the relevant information of the transformations needs to be stored. We display the cost of zeroing all the elements, except the first one, of the first column of an $m \times n$ matrix. This amounts to one Householder reflection or $m - 1$ Givens rotations.

- Storage:
  - Householder reflections: one m-vector $\boldsymbol{u}$, one scalar $\boldsymbol{u}^T\boldsymbol{u}$.
  - $m-1$ Givens rotations: one scalar per rotation, giving a total of $m-1$ scalars.
- Work:
  - Householder reflections: $2m$ flops and 1 square root to compute $\boldsymbol{u}$ and zero all elements below the first entry of an m-vector. Plus $2m(n-1)$ flops for applying it to the rest of the $m \times n$ matrix.
  - $m-1$ Givens rotations: $5(m-1)$ flops and $m-1$ square roots to compute the rotations. Plus $6n \times (m-1)$ flops to apply them to the $m \times n$ matrix.

All the transformations have very good numerical behavior, including fast Givens. They are backward stable, i.e., the numerical application of an orthogonal transformation is equivalent to an exact transformation of a slightly perturbed vector, with a relative perturbation of the size of the machine precision.

QR factorization using Householder or Givens transformations

Using Householder or Givens transformations one can obtain a QR factorization with a square $m \times m$ matrix Q. The solution steps are as follows:

1. Successive application of either $n-1$ Householder reflections or $(m-1)$, $(m-2)$, $\ldots$, $(m-n)$ Givens rotations, to zero successive columns of A below the diagonal:
   - $A^{(1)} = A$ and for $k = 1, \ldots, n-1$:
   - either $A^{(k+1)} = H_k A^{(k)}$ or $A^{(k+1)} = G_k A^{(k)}$.

   Here, H_k and $G_k = G^{(k+1)} \cdots G^{(m)}$ denote the Householder transformation and the product of Givens transformations that are necessary to zero the elements below the diagonal in the kth column of $A^{(k)}$. Hence
   - $Q^T = H_{n-1} \cdots H_1$ or $Q^T = G_{n-1} \cdots G_1$.
2. The right-hand side must also be transformed, i.e., we must compute
   - $\boldsymbol{d} = \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{pmatrix} = H_{n-1} \cdots H_1 \boldsymbol{b}$ or $\boldsymbol{d} = \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{pmatrix} = G_{n-1} \cdots G_1 \boldsymbol{b}$.

3. Solution of the equivalent LSQ problem and computation of the residual norm:

   - $\boldsymbol{x}^* = R_1^{-1}\boldsymbol{d}_1, \quad \|\boldsymbol{r}^*\|_2 = \|\boldsymbol{d}_2\|_2.$

4. Optional: Compute a condition number estimate via R_1; see [20] p. 114 ff.

The matrix Q is stored in factorized form. To save memory, this can be done by overwriting the matrix A with Q and R, plus using some small additional storage only. The total amount of work is $2n^2(m - \frac{1}{3}n)$ flops for Householder and $3n^2(m - \frac{1}{3}n)$ flops for Givens, for a full matrix. The Givens method is more efficient in the structured sparse case, for example, when applied to a Hessenberg matrix (an upper Hessenberg matrix has zero entries below the first subdiagonal and a lower Hessenberg matrix has zero entries above the first superdiagonal).

It is proved in [105] that the computed LSQ solution has an approximate relative error bounded by $\varepsilon_M[\mathrm{cond}(A) + \|\boldsymbol{r}^*\|_2\mathrm{cond}(A)^2]$. Any ill-conditioning of A will appear as ill-conditioning of the triangular matrix R_1 of the QR factorization, because $\mathrm{cond}(R_1) = \mathrm{cond}(A)$. Therefore, both Householder and Givens QR factorization algorithms break down in the back-substitution if $\mathrm{cond}(A) \simeq 1/\varepsilon_M$, as shown by the following example computed in MATLAB with machine precision $\varepsilon_M \simeq 2.22 \cdot 10^{-16}$.

Example 61. *Define the* 3×2 *matrix*

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 - 10^{-16} \end{pmatrix},$$

with condition number $\mathrm{cond}(A) = 3.78 \cdot 10^{16}$. *Using Householder transformations to compute the QR factorization we obtain the triangular matrix*

$$R_1 = \begin{pmatrix} -1.7321 & -1.7321 \\ 0 & -9.1575 \cdot 10^{-17} \end{pmatrix},$$

which is essentially singular, revealing that $A\boldsymbol{x} \approx \boldsymbol{b}$ *should be treated as a rank-deficient problem.*

Variants of Gram-Schmidt orthogonalization

Given a set of linearly independent vectors $\{\boldsymbol{a}_k\}_{k=1}^n$ in $\mathbb{R}^m$, the classical Gram-Schmidt (GS) process generates an orthonormal set of vectors $\{\boldsymbol{q}_k\}_{k=1}^n$ spanning the same subspace. The $\boldsymbol{q_k}$ are generated successively by subtracting from each $\boldsymbol{a}_k$ the components in the directions of already obtained orthonormal vectors. While being an important theoretical tool and

a computationally advantageous algorithm, because it is well adapted for parallel computation (the main work can be performed as a matrix-vector multiplication), numerically GS often produces a non-orthogonal set of vectors due to cancellations in the substractions. A mathematically equivalent and more robust variant of GS, although not as convenient for parallelization, is the modified Gram-Schmidt (MGS) orthogonalization process:

Modified Gram-Schmidt

for $k = 1, \ldots, n$
  $r_{kk} = \|\boldsymbol{a}_k\|_2$
  $\boldsymbol{q}_k = \boldsymbol{a}_k / r_{kk}$
  for $j = k+1, \ldots, n$
    $r_{kj} = (\boldsymbol{a}_j^T \boldsymbol{q}_k)$
    $\boldsymbol{a}_j = \boldsymbol{a}_j - r_{kj}\, \boldsymbol{q}_k$
  end
end

In this algorithm the process of orthogonalizing a vector $\boldsymbol{a}_k$ is refined successively, i.e., $\boldsymbol{q}_k$ is obtained by first projecting $\boldsymbol{a}_k$ onto the subspace orthogonal to $\boldsymbol{q}_1$, then the *resulting* vector is projected onto the subspace orthogonal to $\text{span}\{\boldsymbol{q}_1, \boldsymbol{q}_2\}$, up to the projection onto the subspace orthogonal to $\text{span}\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{k-1}\}$. This avoids an amplification of the rounding errors affecting the orthogonality of previously computed $\boldsymbol{q}_k$, which may happen in classical GS. This amplification may also occur in MGS, but to a lesser extent, depending on the linear independence of the original vectors, i.e., on $\text{cond}(A)$ (details can be found in the experimental work of [26], [136] or [208]). Björck proved that for MGS the loss of orthogonality can be bounded by $\|I - Q_1^T Q_1\|_2 \leq \text{cond}(A)\varepsilon_M$, whereas there is no such bound for the GS algorithm. In any case, if one needs a highly accurate orthogonal set, one can always *reorthogonalize* with MGS, a process that needs only to be done once, as described in [20].

The algorithm requires $2mn^2$ flops, as it always manipulates vectors of size m. To save storage, A can be overwritten with $Q_1 = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n]$, but R_1 needs extra storage. The process can be written in matrix form as right multiplication of A by the square $n \times n$ upper triangular matrices $R^{(1)}, R^{(2)}, \ldots, R^{(n)}$, such that $A\, R^{(1)}\, R^{(2)} \cdots R^{(n)} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n]$, with the $R^{(i)}$ matrices given by

$$R^{(1)} = \begin{pmatrix} \frac{1}{r_{11}} & -\frac{r_{12}}{r_{11}} & \cdots & -\frac{r_{1n}}{r_{11}} \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix},$$

$$R^{(2)} = \begin{pmatrix} 1 & & & & \\ & \frac{1}{r_{22}} & -\frac{r_{23}}{r_{22}} & \cdots & -\frac{r_{2n}}{r_{22}} \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}, \ldots, R = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & & \frac{1}{r_{nn}} \end{pmatrix}.$$

Here the r_{ij} are the elements of the current transformed A matrix. Thus, the MGS algorithm can be used to produce the economical QR factorization of $A = Q_1 R_1$ with $R = (R^{(1)}\, R^{(2)} \cdots R^{(n)})^{-1} = (R^{(n)})^{-1} \cdots (R^{(1)})^{-1}$. This factorization generates at the kth step the kth column of Q_1 and the kth row of R_1. This is another useful aspect of MGS: it produces vectors of the new orthonormal basis from the first iteration, whereas in order to obtain any basis vectors from the Householder or Givens transformations one has to wait until completion, i.e., until all m columns of the square matrix Q have been computed.

In order to apply MGS when the rank is unknown, one can improve the basic algorithm by adding column pivoting, as will be explained in detail in the next chapter. The following example from [20] illustrates the superiority of MGS to classical GS.

Example 62. *Consider the matrix*

$$A = \begin{pmatrix} 1 & 1 & 1 \\ \varepsilon & & \\ & \varepsilon & \\ & & \varepsilon \end{pmatrix}, \qquad \varepsilon \text{ small enough so that } fl(1+\varepsilon^2) = 1.$$

Then the Q_1 matrices obtained through GS and MGS are

$$Q_{\mathrm{GS}} = \begin{pmatrix} 1 & & \\ \varepsilon & -\alpha & -\alpha \\ & \alpha & \\ & & \alpha \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & & \\ \varepsilon & -\alpha & -\beta/2 \\ & \alpha & -\beta/2 \\ & & \beta \end{pmatrix},$$

with $\alpha = 2^{-1/2}$, $\beta = (\frac{2}{3})^{1/2}$. The maximum deviation from orthogonality for GS is $|\boldsymbol{q}_2^T \boldsymbol{q}_3| = 1/2$ and for MGS $|\boldsymbol{q}_1^T \boldsymbol{q}_3| = \sqrt{2/3}\,\varepsilon$, where ϵ could be as large as $\sqrt{\epsilon_M} \approx 10^{-8}$.

QR factorization using modified Gram-Schmidt

In principle, once the economical QR factorization $A = Q_1 R_1$ is computed using MGS, one could obtain the solution $\boldsymbol{x}^*$ of the least squares problem by solving the triangular system $R_1 \boldsymbol{x} = Q_1^T \boldsymbol{b}$. However, this may not give an accurate solution for an ill-conditioned matrix due to the potential loss of orthogonality in the MGS process, as shown above. A reorthogonalization

would give an accurate solution while duplicating the cost. A more stable algorithm is obtained if one instead applies MGS to the bordered matrix:

$$\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} = \begin{pmatrix} Q_1 & \boldsymbol{q}_{n+1} \end{pmatrix} \begin{pmatrix} R_1 & \boldsymbol{z} \\ 0 & \varrho \end{pmatrix}.$$

Then we have

$$\|A\,\boldsymbol{x} - \boldsymbol{b}\|_2 = \left\| \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} \right\|_2 = \left\| Q_1\,(R_1\,\boldsymbol{x} - \boldsymbol{z}) - \varrho\,\boldsymbol{q}_{n+1} \right\|_2.$$

Since $\boldsymbol{q}_{n+1}$ is orthogonal to Q_1, because it was produced by the MGS process, the term $\varrho\,\boldsymbol{q}_{n+1}$ is not taken into consideration in the minimization process and the LSQ solution is therefore:

$$\boldsymbol{x}^* = R_1^{-1}\,\boldsymbol{z}, \qquad \|\boldsymbol{r}^*\|_2 = \|\varrho\,\boldsymbol{q}_{n+1}\|_2 = |\varrho|.$$

For more details see [16, 20, 105]. Note that it is not necessary to assume that Q_1 is perfectly orthogonal.

MGS has been proved to be backward stable by interpreting the MGS decomposition as the Householder QR factorization applied to the matrix A augmented with a square $n \times n$ matrix of zeros on top:

$$\begin{pmatrix} 0 \\ A \end{pmatrix}.$$

For details see [26].

One final point to stress is, quoting from Peters and Wilkinson (see [20], p. 66): "Evidence is accumulating that the modified Gram-Schmidt method (applied to the augmented system) gives better results than Householder." Experimental evidence shows that the LSQ solutions obtained via MGS have roughly 15% more correct figures. The reason seems to be related to the comparative invariance of MGS to row permutations. In fact, Björck and Paige [26] state: "If the matrix is not well row scaled, row interchanges may be needed to give an accurate solution for the least squares problem. In this context, it is interesting to note that MGS is numerically invariant under row permutations of A. However if row interchanges are used also in QR and the row norms of A vary widely, QR is less expensive because MGS would need reorthogonalization."

Example 63. *Figure 4.3.1 compares the accuracy of most of the LSQ algorithms covered so far, namely, normal equations, LU factorization (Peters and Wilkinson algorithm), Householder QR factorization, MGS and MGS applied to the bordered matrix* $(A, \boldsymbol{b})$. *The test matrices are the same as in*

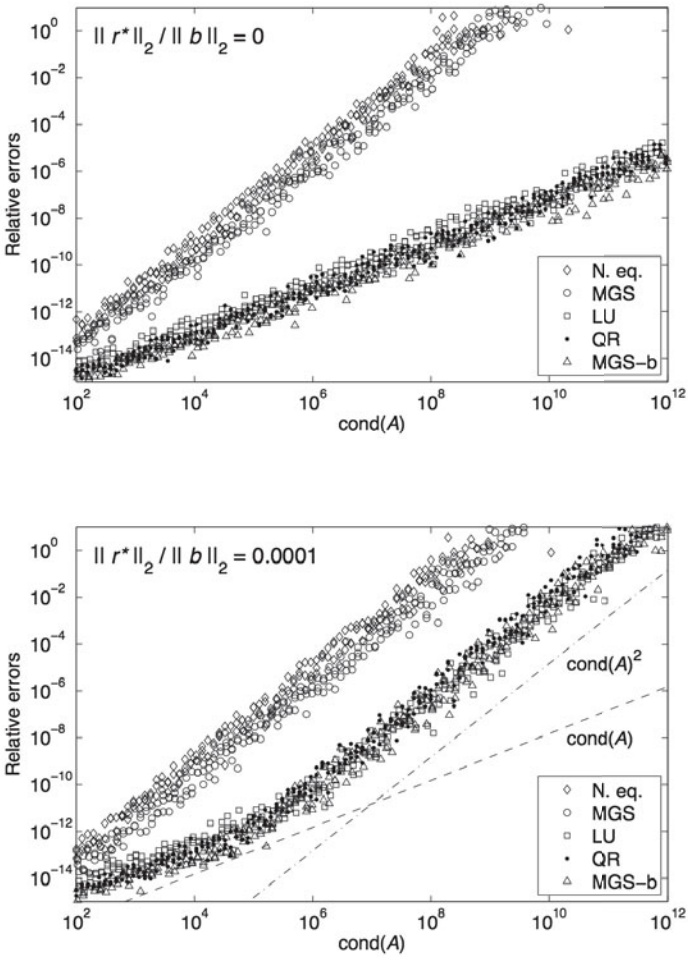

Figure 4.3.1: The relative errors of LSQ solutions as functions of cond(A) computed by means of the following methods: N. eq. = normal equations, MGS = modified Gram-Schmidt, LU = LU factorization, QR = Householder QR factorization and MGS-b = MGS applied to the bordered matrix $(A \quad \boldsymbol{b})$. We show errors for problems with zero/negligible residual (top) and a larger residual (bottom). The last three methods are always superior to the first two, and notice in the bottom plot that the errors change from being proportional to cond(A) for well-conditioned problems to being proportional to cond$(A)^2$ for ill-conditioned ones. The top plot is for an LSQ problem with zero residual, while the bottom plot is for a problem with nonzero residual.

Example 58, and we show the relative errors as functions of $\mathrm{cond}(A)$*. This example shows that the errors from the normal equations and the "naive" MGS approach are proportional to* $\mathrm{cond}(A)^2$ *independently of the size of the residual.*

The errors from the other three approaches, on the other hand, depend on the size of the residual. For problems with a small/negligible residual, the errors are proportional to $\mathrm{cond}(A)$*, as seen in the top plot. For problems with a larger residual, the behavior of the errors depend on the size of* $\mathrm{cond}(A)$*, as seen in the bottom plot. Notice the bend of the error curve at* $\mathrm{cond}(A) \approx 10^3$*. For well-conditioned problems, the errors are proportional to* $\mathrm{cond}(A)$*, while they are proportional to* $\mathrm{cond}(A)^2$ *for ill-conditioned problems, as predicted by the theory. This correctly matches the sensitivity of least squares solutions to perturbations in the data and confirms the stability of the algorithms.*

The above example illustrates that LU and QR factorization and MGS applied to the bordered matrix are preferable when accuracy matters. The example also illustrates that the error analysis from Section 3.5 carries over to the influence of rounding errors, meaning that for ill-conditioned problems with large residual we see a dependence on $\mathrm{cond}(A)^2$ for all three accurate methods.

4.4 Modifying least squares problems

In many applications, a number of closely related least squares problems have to be solved. For example, there might be some errors in the matrix and therefore one might want to change some of its elements. In time series analysis, such as in the neural network application of Section 12.1, a *data window* slides in time, deleting old data and adding new data, i.e., changing the rows of the data matrix (down- and up-dating). Or in regression analysis, one checks if a variable is representative by adding or eliminating it from the model used, i.e., changing columns. Additional uses appear in some of the constrained LSQ algorithms. Therefore, we will consider how a matrix factorization is changed under two possible types of modifications: adding/deleting rows (i.e., data) and adding/deleting columns (i.e., variables).

An important point in the design of modification procedures is the efficiency of the algorithms, because often the process has to be done in real time. In addition, stability is essential because, for efficiency the computations are often performed in single precision.

One of the first considerations when modifying a least squares problem with full-rank matrix A is to check if the modified matrix $\overline{A}$ remains of full rank. Based on Theorem 4 from Appendix B about interlacing singular

	add	delete
column	$\overline{A}$ may be singular	$\overline{A}$ is full-rank
row	$\overline{A}$ is full-rank	$\overline{A}$ may be singular

Table 4.1: Consequences of modifying a matrix A to a new matrix $\overline{A}$ by adding or deleting a row or column.

values, the consequences of adding or deleting rows or columns of A are summarized in Table 4.1.

Although updating algorithms have been designed for a number of matrix factorizations like LU, Cholesky and complete orthogonal decompositions, we will not consider them here, but refer to [20, 87] for an exhaustive treatment and rather concentrate on the less expensive algorithms, i.e., describe how to modify the normal equations and the QR factorization of a full-rank matrix.

Normal equations (recursive least squares)

The algorithm known as *recursive least squares* is used in many applications (signal analysis, for example), due to its simplicity and recursive nature. It is also known as the *covariance matrix method* because it modifies the inverse of the normal equations matrix $\left(A^T A\right)^{-1}$, which is the unscaled covariance matrix. Unfortunately, it is very sensitive to rounding errors.

Given the normal equations formulation of the least squares problem, assume that a new observation $\boldsymbol{w}^T \boldsymbol{x} = \beta$ is added. The updated solution $\bar{\boldsymbol{x}}^*$ satisfies the normal equations system,

$$\left(A^T A + \boldsymbol{w}\,\boldsymbol{w}^T\right) \bar{\boldsymbol{x}}^* = A^T \boldsymbol{b} + \beta \boldsymbol{w}.$$

This solution can be obtained via the Sherman-Morrison-Woodbury formula (see B.4) and the inverse of $A^T A$,

$$\bar{\boldsymbol{x}}^* = \boldsymbol{x}^* + \frac{\beta - \boldsymbol{w}^T \boldsymbol{x}^*}{1 + \boldsymbol{w}^T \boldsymbol{u}}\, \boldsymbol{u}, \qquad \boldsymbol{u} = \left(A^T A\right)^{-1} \boldsymbol{w}.$$

The inverse is really not necessary since $\boldsymbol{u}$ can be computed via the Cholesky factor of $A^T A$. If, instead, an observation is deleted, then the updated solution $\bar{\boldsymbol{x}}$ satisfies the system

$$\left(A^T A - \boldsymbol{w}\,\boldsymbol{w}^T\right) \bar{\boldsymbol{x}}^* = A^T \boldsymbol{b} - \beta \boldsymbol{w}$$

and the updated solution is

$$\bar{\boldsymbol{x}}^* = \boldsymbol{x}^* - \frac{\beta - \boldsymbol{w}^T \boldsymbol{x}^*}{1 - \boldsymbol{w}^T \boldsymbol{u}}\, \boldsymbol{u},$$

with the same $\boldsymbol{u}$ as before. It is of course assumed in both cases that the updated least squares problem continues to have a full-rank matrix.

Full QR decompositions

Even without pivoting, the QR factorization algorithm is stable, and it is used extensively in applications that require updating. For simplicity we will consider changes that involve adding/deleting only one column or row, but the algorithms given can easily be modified for changes involving blocks of rows or columns [70]. A good reference for the material in this subsection is chapter 12 of [105]. We note that the updating algorithms can also be used in connection with QR factorization of the bordered matrix $(\ A \quad \boldsymbol{b} \ )$; cf. Theorem 56.

As seen from Table 4.1, the modified matrix may have a different rank and in particular it may become rank deficient. Because using the SVD and then updating is too expensive (the amount of work involved being comparable to an SVD decomposition from scratch), QR-based decompositions such as ULV and URV are used for this purpose. See [20] and [78] for more details.

For the row modification algorithms we will assume, for simplicity, that the first row is involved. When instead a row is added/deleted in an arbitrary position, the procedure is similar but involves some permutations.

Adding a row. Assume that, starting with $A = Q\,R$, we wish to obtain the QR factorization of $\overline{A} = \begin{pmatrix} \boldsymbol{w}^T \\ A \end{pmatrix}$. Note that

$$\begin{pmatrix} 1 & 0 \\ 0 & Q^T \end{pmatrix} \bar{A} = \begin{pmatrix} \boldsymbol{w}^T \\ R \end{pmatrix} = \overline{H},$$

with $\overline{H}$ an $(m+1) \times n$ upper Hessenberg matrix. Using Givens rotations $G_1, \ldots, G_n$ to zero the elements of R from r_{11} onward until r_{nn} will transform $\overline{H}$ to an upper triangular matrix $\overline{R}$. The decomposition of $\overline{A}$ is

$$\overline{A} = \begin{pmatrix} 1 & 0 \\ 0 & Q^T \end{pmatrix} G_1^T \cdots G_n^T \, \overline{R}.$$

The number of operations involved in the update is $3n^2 + \mathcal{O}(n)$ flops.

Deleting a row. Deleting a row is somewhat more involved. Given an $m \times n$ matrix $A = QR$, one wants the decomposition of $\overline{A}$, the lower part of A:

$$A = \begin{pmatrix} \boldsymbol{z}^T \\ \overline{A} \end{pmatrix}$$

The algorithm has the following steps:

- Use $m-1$ Givens rotations $G_1, \ldots, G_{m-1}$ such that when applied to the first row $\boldsymbol{q}^T$ of Q one obtains $G_1^T ... G_{m-1}^T \boldsymbol{q} = \pm \widehat{\boldsymbol{e}}_1$. Then,

$$Q\, G_{m-1} ... G_1 = \begin{pmatrix} \pm 1 & 0 \\ 0 & \overline{Q} \end{pmatrix},$$

  with $\overline{Q}$ orthogonal and dimensions $(m-1) \times (m-1)$.

- When the Givens rotations are applied to R one obtains an upper Hessenberg matrix

$$\overline{H} = G_1^T \cdots G_{m-1}^T R = \begin{pmatrix} \boldsymbol{v}^T \\ \overline{R} \end{pmatrix}$$

  and

$$\begin{aligned} A &= \begin{pmatrix} \boldsymbol{z}^T \\ \bar{A} \end{pmatrix} = (Q G_{m-1} \cdots G_1)(G_1^T \cdots G_{m-1}^T R). \\ &= \begin{pmatrix} \pm 1 & 0 \\ 0 & \bar{Q} \end{pmatrix} \begin{pmatrix} \boldsymbol{v}^T \\ \overline{R} \end{pmatrix} \end{aligned}$$

  Therefore $\overline{A} = \overline{Q}\, \overline{R}$.

Here the matrix Q is needed to determine the triangular matrix $\overline{R}$. The number of flops is $\mathcal{O}(m^2)$, so if n is small this procedure is not worthwhile, and instead it is better to compute the QR factorization from scratch.

Adding a column. Starting from the initial factorization

$$A = (\, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \,) = Q\, R,$$

assume that a new column $\boldsymbol{z}$ is added,

$$\overline{A} = (\, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_k, \boldsymbol{z}, \boldsymbol{a}_{k+1}, \ldots, \boldsymbol{a}_n \,).$$

The following two steps are needed:

- First compute the vector $\boldsymbol{w} = Q^T \boldsymbol{z}$. In $Q^T \bar{A}$, one has then a matrix that is essentially R, except for the full vector $\boldsymbol{w}$ after column k:

$$Q^T \bar{A} = \left(Q^T \boldsymbol{a}_1, ..., Q^T \boldsymbol{a}_k, \boldsymbol{w}, Q^T \boldsymbol{a}_{k+1}, ..., Q^T \boldsymbol{a}_n\right) \equiv B,$$

  where the matrix B has the form

$$B = \begin{pmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & 0 & x \\ 0 & 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & x & 0 & 0 \end{pmatrix}.$$

- Now apply Givens rotations, $G_{m-1}, \ldots, G_{k+1}$ to obtain an upper triangular structure. Note though, that in the process of zeroing elements $w_{k+2}, \ldots, w_n$ some fill-in will occur in $Q^T \boldsymbol{a}_{k+1}, \ldots, Q^T \boldsymbol{a}_n$, but without marring the triangular structure. For the above B the steps are

$$\begin{pmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & 0 & x \\ 0 & 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & x & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & 0 & x \\ 0 & 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow$$

$$\begin{pmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & 0 & x \\ 0 & 0 & 0 & 0 & 0 & x \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & 0 & x \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Again, the original matrix Q is needed to compute the new triangular matrix. The work is done in $\mathcal{O}(mn)$ flops.

Deleting a column.
We start with the decomposition of $A = (\, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \,) = QR$ with $R = (\, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_n \,)$. Assume that the kth column of A is deleted,

$$\overline{A} = (\, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{k-1}, \boldsymbol{a}_{k+1}, \ldots, \boldsymbol{a}_n \,).$$

It is easy to prove that

$$\bar{A} = Q \; (\, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_{k-1}, \boldsymbol{r}_{k+1}, \boldsymbol{r}_n) \equiv Q \, \overline{H},$$

where $\overline{H}$ is R with the kth column removed; thus it is triangular up to column $k-1$ and has a Hessenberg structure from there on, e.g., for $m = 7$, $n = 5$, $k = 3$,

$$\overline{H} = \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

To obtain a QR factorization of $\overline{A}$, a single Givens transformation has to be applied to each of the columns of $\overline{H}$ from $\bar{\boldsymbol{h}}_{k+1}, \ldots, \bar{\boldsymbol{h}}_n$, to zero the elements $\bar{h}_{k+1,k}, \ldots, \bar{h}_{n,n-1}$, i.e., $\overline{R} = G_{n-1}^T \cdots G_k^T \overline{H} \Leftrightarrow \overline{H} = G_k \cdots G_{n-1} \overline{R}$. The new QR factorization is therefore

$$\bar{A} = (Q\, G_k \cdots G_{n-1})\, \overline{R} \equiv \overline{Q}\, \overline{R}.$$

The work required is $\mathcal{O}(n^2)$ flops. Note that we do not make use of the matrix Q to compute the triangular factor $\overline{R}$.

4.5 Iterative refinement

The LSQ solution computed via the normal equations or the QR factorization can be improved by iterative procedures that use the already available factorization, although they need some additional storage. One should be careful though if the problem is too ill-conditioned, as it does not make sense to compute a very accurate solution to such a problem when the data have comparatively large errors. There are several techniques that are in use, differing in convergence rates and applicability. The first that we will describe is the natural extension to least squares problems of the iterative refinement method for linear systems.

Iterative refinement as extension of the linear system case

Given an initial approximation $\widetilde{\boldsymbol{x}}^*$ to the LSQ solution, with residual $\widetilde{\boldsymbol{r}}^* = \boldsymbol{b} - A\,\widetilde{\boldsymbol{x}}^*$, the error $\Delta\boldsymbol{x}$ is defined as $\widetilde{\boldsymbol{x}}^* + \Delta x = \boldsymbol{x}^*$ and it follows that

$$\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2 = \min_{\Delta\boldsymbol{x}} \left\|\boldsymbol{b} - A\,(\widetilde{\boldsymbol{x}}^* + \Delta\boldsymbol{x})\right\|_2 = \min_{\Delta\boldsymbol{x}} \left\|\widetilde{\boldsymbol{r}}^* - A\,\Delta\boldsymbol{x}\right\|_2 .$$

Hence, the correction $\Delta\boldsymbol{x}$, in theory, can be obtained from solving a new LSQ problem with the original matrix A. The correcting step may have to be repeated if the new approximate solution is still not satisfactory. The residual should be computed in *extended precision* and then rounded to *standard (fixed) precision.*

This algorithm, implemented using the QR factorization, was analyzed in [107], where it is shown that the procedure for solution refinement is adequate only if the LSQ problem is nearly consistent.

Iterative refinement using the augmented system

A second, more robust and successful iterative refinement algorithm, using extended precision where necessary, corrects not only the solution but also the residual [17, 18]. The appropriate formulation of this least squares

problem is the augmented system (2.1.4). The errors $\Delta \boldsymbol{x} = \boldsymbol{x}^* - \widetilde{\boldsymbol{x}}^*$ and $\Delta \boldsymbol{r} = \boldsymbol{r}^* - \widetilde{\boldsymbol{r}}^*$ satisfy

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \widetilde{\boldsymbol{r}}^* + \Delta \boldsymbol{r} \\ \widetilde{\boldsymbol{x}}^* + \Delta \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix},$$

leading to the solution of the system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{r} \\ \Delta \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix} - \begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \widetilde{\boldsymbol{r}}^* \\ \widetilde{\boldsymbol{x}}^* \end{pmatrix} \equiv \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix}$$

to obtain the corrections. The algorithm steps are

- Choose as initial values $\begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{x}_0 \end{pmatrix} = \begin{pmatrix} \widetilde{\boldsymbol{r}}^* \\ \widetilde{\boldsymbol{x}}^* \end{pmatrix}$.
- For $k = 0, 1, \ldots$ until convergence, compute the "residuals" for the augmented system using *extended precision* for the inner products,

$$\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix} - \begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{r}_k \\ \boldsymbol{x}_k \end{pmatrix}.$$

- Using the economical QR factorization of A, multiply the first equation by Q_1^T and use $A^T = R_1^T Q_1^T$ to obtain the system

$$\begin{pmatrix} I & R_1 \\ R_1^T & 0 \end{pmatrix} \begin{pmatrix} Q_1^T \Delta \boldsymbol{r} \\ \Delta \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} Q_1^T \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix}.$$

  Solve this system in *standard precision*.

- Compute $\Delta \boldsymbol{r} = Q_1 \left(Q_1^T \Delta \boldsymbol{r} \right)$ and correct the solution and the residual:

$$\begin{pmatrix} \boldsymbol{r}_{k+1} \\ \boldsymbol{x}_{k+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_k \\ \boldsymbol{x}_k \end{pmatrix} + \begin{pmatrix} \Delta \boldsymbol{r} \\ \Delta \boldsymbol{x} \end{pmatrix}.$$

Each iteration involves the solution of two triangular systems and a premultiplication by $Q_1 Q_1^T$ to obtain $\Delta \boldsymbol{r}$, for a total of $8mn - 2n^2$ flops in standard precision and $4mn$ in extended precision per iteration. The convergence rate here is $c(m,n)\,\mathrm{cond}(A)\varepsilon_M$. In fact, as Björck points out ([20], pp. 123): "Hence, in a sense, iterative refinement is even more satisfactory for large residual least squares problems and may give solutions to full single precision accuracy, even when the initial solution may have no correct significant figures." If one is working with a t-digit binary floating-point number system, then on a problem with condition number $\mathrm{cond}(A) \simeq 2^q$, with $q < t$, after k iterations we can obtain $k(t-q)$ binary digits of accuracy in the LSQ solution. The algorithm works as well if modified Gram-Schmidt is used. Open source software can be found in [64].

Fixed precision iterative refinement and corrected seminormal equations

The last iterative refinement algorithm that we will discuss is a fixed precision iterative refinement method, applied to the so-called seminormal equations

$$R^T R\,\boldsymbol{x} = A^T \boldsymbol{b},$$

where R is the Cholesky factor of $A^T A$.

Fixed precision iterative refinement

$\boldsymbol{x}_0 = \widetilde{\boldsymbol{x}}$
for $k = 0, 1, \ldots$ until convergence
  $\boldsymbol{r}_k = \boldsymbol{b} - A\,\boldsymbol{x}_k$
  solve $R^T(R\,\Delta\boldsymbol{x_k}) = A^T \boldsymbol{r}_k$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x_k}$
end

The cost for each step is roughly $4n^2$ flops (solving triangular systems), plus $4mn$ flops for the two matrix-vector products. Note that the matrix A is needed.

Fixed precision refinement can be used to correct the solution obtained via the normal equations when R is computed from the Cholesky factorization of $A^T A$, although then only a convergence rate of $\mathcal{O}(cond(A)^2)$ can be obtained. For ill-conditioned problems, several iteration steps may be necessary to achieve good accuracy. See section 6.6.5 in [20] for more details.

Another important application of the algorithm is in connection with sparse least squares problems, where it is unfeasible to store the (often quite dense) orthogonal factor Q or Q_1 of the QR factorization of A. To solve a given LSQ problem one only needs to compute and store the triangular factor R_1 and apply the orthogonal transformations "on the fly" to the right-hand side to obtain $Q_1^T \boldsymbol{b}$, and then $\boldsymbol{x}^*$ is obtained via back-substitution. But in order to solve a new system with the same A and a new right-hand side, as is required for iterative refinement, Q_1 would be needed. The immediate solution is to recall that R_1 is a Cholesky factor of the normal-equation matrix $A^T A$, and the seminormal equations could therefore be used. *One* step of the above iterative refinement algorithm produces an improved solution with an error proportional to cond(A). This technique, called *corrected seminormal equations* (CSNE), was derived and analyzed by Björck [19].

4.6 Stability and condition number estimation

In order to obtain accuracy bounds for the error in a computed approximate LSQ solution $\widetilde{\boldsymbol{x}}^*$ to a well-conditioned least squares problem, it is useful to prove the backward stability of the algorithm used (see Appendix A). This implies that $\widetilde{\boldsymbol{x}}^*$ can be expressed as the exact solution of a perturbed least squares problem:

$$\min_x \|(A + \Delta A)\boldsymbol{x} - (\boldsymbol{b} + \Delta \boldsymbol{b})\|_2,$$

where $\|\Delta A\|_2$ and $\|\Delta \boldsymbol{b}\|_2$ are of the order of the machine precision ε_M. In other words, the computed solution "almost" satisfies the least squares minimization criterion $\min_x \|A\,\boldsymbol{x} - \boldsymbol{b}\|_2$.

Given the backward estimate one can then, in principle, estimate the actual error of the computed solution $\widetilde{\boldsymbol{x}}^*$ in terms of the LSQ condition number $\mathrm{cond}(A, \boldsymbol{b})$ of the matrix:

$$\frac{\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2} = \mathcal{O}(\mathrm{cond}(A,\, \boldsymbol{b})\, \varepsilon_M).$$

Hence, a small backward error only implies a computed solution $\widetilde{\boldsymbol{x}}^*$ close to the exact one $\boldsymbol{x}^*$, if the condition number is of reasonable size. The backward stability of the algorithms for full-rank matrices described in this chapter has been studied extensively.

Methods that explicitly form the normal equations cannot be backward stable, because the round-off errors occurring while forming the matrix products are not, in general, perturbations of size ε_M relative to A and $\boldsymbol{b}$. On the other hand, the methods based on the QR factorization are backward stable, including the modified Gram-Schmidt method, provided that the product $Q^T\boldsymbol{b}$ is computed implicitly via the augmented matrix $(\ A \quad \boldsymbol{b}\ )$.

There are many specialized methods that apply to structured LSQ problems, such as those with a Toeplitz matrix, which are not – or are not known to be – backward stable. Given a computed $\widetilde{\boldsymbol{x}}^*$, it is possible to verify numerically whether an algorithm is stable by computing an estimate for the smallest backward error. In fact, a theorem proved in [249] gives an expression for the optimal backward error in the Frobenius norm, once an approximate least squares solution has been computed. For practical purposes, less expensive bounds are presented in [21], p. 6. One can be satisfied with the computed solution if the backward errors ΔA and $\Delta \boldsymbol{b}$ are small in comparison to the uncertainties in the data.

As mentioned in Section 3.5 on perturbation analysis, an important factor in the behavior of the solution to the least squares problem is the matrix condition number $\mathrm{cond}(A, \boldsymbol{b})$. It is therefore useful to obtain an estimate for this number, if possible as a by-product of the computations

Method	Flops	Sensitivity	Backwd stab
Normal equations	$mn^2 + \frac{1}{3}n^3$	$\text{cond}(A)^2$	No
LU Cholesky	$2mn^2 - \frac{2}{3}n^3$	$\text{cond}(L)^2$	No
LU Cline	$3mn^2 - \frac{7}{3}n^3$	$\text{cond}(A)$	Yes
QR Householder	$2mn^2 - \frac{2}{3}n^3$	$\text{cond}(A, \boldsymbol{b})$	Yes
QR Givens	$3mn^2 - n^3$	$\text{cond}(A, \boldsymbol{b})$	Yes
QR fast Givens	$2mn^2 - \frac{2}{3}n^3$	$\text{cond}(A, \boldsymbol{b})$	Yes
QR MGS on $(A, \boldsymbol{b})$	mn^2	$\text{cond}(A, \boldsymbol{b})$	Yes

Table 4.2: Comparison of the different computational methods for solving the LSQ problem from [105].

needed in the least squares solution process. This is obviously possible in the specific case of the SVD algorithm, since the singular values are computed explicitly. If the QR factorization is used and the matrix has full rank, there are two options, both exploiting the triangular form of R_1.

One method, applicable if the QR factorization algorithm with *column pivoting* is used, is based on the fact that there is a lower estimate for $\text{cond}(A)$ from the inequalities

$$|r_{11}| \leq \sigma_1 = \|R_1\|_2 \leq n^{1/2}|r_{11}| \qquad \text{and} \qquad \sigma_n^{-1} = \|R_1^{-1}\|_2 \geq |r_{nn}^{-1}|.$$

Using these results one obtains a lower bound for the condition as

$$|r_{11}|/|r_{nn}| \leq \text{cond}(A).$$

This may be a severe underestimate, though. One can improve it by performing inverse iteration with $R_1^T R_1$ to obtain a good estimate for the dominant eigenvalue of $(R_1^T R_1)^{-1}$ and thus for $1/\sigma_n$. This can even be used if the matrix is *nearly* rank deficient.

Another condition estimator, first designed by Hager, computes the condition number of a triangular matrix in the L_1 or L_∞ norm, using optimization techniques. A bound for an estimate of the spectral condition number can then be obtained from

$$\|A\|_2 \leq (\|A\|_1 \, \|A\|_\infty)^{\frac{1}{2}}.$$

Details can be found in [20] and the references cited there.

4.7 Comparison of the methods

Table 4.2 compiles a list of properties for the computational methods discussed in this section. The second column shows the dominant term in the

flop counts, i.e., the terms $\mathcal{O}(mn)$ and $\mathcal{O}\left(n^2\right)$ are not included. The third column shows the sensitivity to rounding errors, and the fourth column shows whether the algorithm is backward stable. All the methods are storage efficient, since they can essentially overwrite the matrix A. If iterative refinement is to be used, however, then A must be saved.

- **Normal equations.** In spite of the squared condition number, the normal equations should not be discarded out of hand because they require less storage and only about half the number of operations compared to the QR factorization methods. The method breaks down for $\text{cond}(A) \approx \varepsilon_M^{-1/2}$ or greater.
- **LU factorization.** The Peters-Wilkinson LU factorization method is a good choice if $m \simeq n$. This method can also be used to produce a better conditioned problem, sometimes just by using a partial factorization.
- **QR factorization.** The QR factorization methods are the most robust. QR using Householder- or Givens-type orthogonalizations only breaks down when $\text{cond}(A) \approx \varepsilon_M^{-1}$. The QR (MGS) method produces the economical QR decomposition, whereas the Householder and Givens forms produce the full version. Givens transformations are well suited for sparse problems.
- **Iterative refinement.** Although one can apply iterative refinement to both normal equations and the QR method, the rate of convergence for the former includes $\mathcal{O}(\text{cond}(A)^2)$, while for the latter includes only $\mathcal{O}(\text{cond}(A))$.

Some final comments regarding the sensitivity to rounding errors of the methods discussed so far, for the full-rank case: if $\text{cond}(A)$ is large and the residual norm $\|\boldsymbol{r}^*\|_2$ is small, i.e., the equations are almost consistent, then QR (Householder or Givens) is more accurate than the normal equations. If $\text{cond}(A)$ and the residual norm are both large, then all methods are inaccurate!

Chapter 5

Direct Methods for Rank-Deficient Problems

The methods developed so far break down when the matrix A is rank deficient, i.e., for $\text{rank}(A) = r < n$, in which case there is no unique solution. Within the linear manifold of solutions, the minimum and basic solutions are specially useful because:

- If we assume that A is derived from noisy data, then the minimum-norm solution $\widehat{\boldsymbol{x}}^* = \sum_{k=1}^{r} \frac{\boldsymbol{u}_k^T \boldsymbol{b}}{\sigma_k} \boldsymbol{v}_k$ will filter the small singular values corresponding to noise (see Theorem 44).
- If the rank deficiency of A is caused by a redundancy in the terms of the model, a basic solution $\boldsymbol{x}_{\text{B}}$ will choose only linearly independent columns of A, i.e., this approach provides subset selection.
- Also, there is currently a strong interest in sparse solutions (compressed sensing) and the basic solution can provide a starting point when seeking even sparser solutions.
- There are other applications where the minimum norm solution is of importance, such as in solving nonlinear problems by successive linearizations, where we want to keep the size of the corrections as small as possible.

This chapter gives a short introduction to numerical methods designed specifically for rank-deficient problems. They are also appropriate when, as is often the case, there is no a priori knowledge of the rank. We start with an analysis of the rank issue, then describe some factorization algorithms that can handle rank deficiency: LU and QR with column permutations and complete orthogonal factorization methods, including the URV, ULV and

VSV decompositions. Finally, we introduce bidiagonal reductions as part of the SVD algorithm. Some of the algorithms are more appropriate for computing basic solutions, others for computing minimum-norm solutions.

5.1 Numerical rank

Knowledge of the rank of A is useful when computing a solution to an LSQ problem. However, in the presence of limited accuracy arithmetic it is not enough to know the mathematical rank of A. In fact, an $m \times n$ matrix with $m \geq n$ may have full rank mathematically – meaning that all n singular values are positive – but if some of them are very small, for example $\sigma_n \simeq \varepsilon_M$, then, for all computational purposes, the matrix is rank deficient. Or, vice versa, a matrix with mathematical rank $r \leq \min(m, n)$, whose elements have been perturbed by a small amount, could change rank. We therefore introduce the concept of *numerical rank*, depending on a given error level ε.

Definition 64. *The numerical ε-rank r_ε of a matrix A is the minimum rank of all possible norm-ε perturbations of A:*

$$\boxed{r_\varepsilon \equiv \min_{\|E\|_2 \leq \varepsilon} \operatorname{rank}(A + E).} \tag{5.1.1}$$

Expressed in another way, r_ε is the number of columns of A that remain linearly independent, assuming a given error level ε for perturbations of the matrix.

A quantitative definition can be obtained using the singular value decomposition. In fact, using the bounds obtained in Theorem 41 for the singular values of a perturbation of A, one can see that the singular values $\sigma_i(A + E)$ of the perturbed matrix can be considered *numerically zero* if $\sigma_i(A + E) \leq \varepsilon$. This justifies the following determination of the numerical rank. The index r_ε is the numerical ε-rank if

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{r_\epsilon} > \varepsilon \geq \sigma_{r_\epsilon+1} \geq \ldots \geq 0. \tag{5.1.2}$$

In other words, given the tolerance ε, the singular values from $\sigma_{r_\varepsilon+1}$ onward are indistinguishable from zero. This criterion for rank determination should be used only if there is a clear gap between the consecutive singular values σ_{r_ε} and $\sigma_{r_\varepsilon+1}$, because the number r_ε must be robust with respect to perturbations of the singular values and the threshold.

Often, one considers instead the following criterion that uses the normalized singular values and therefore is scale free:

$$\frac{\sigma_r}{\sigma_1} \leq \tau < \frac{\sigma_{r+1}}{\sigma_1}.$$

This is equivalent to the preceding inequality (5.1.2) with $\tau = \varepsilon/\sigma_1$ and it is more directly related to the condition number of the matrix. In particular, we say that a matrix is numerically rank deficient if $\text{cond}(A) > \varepsilon_M^{-1}$, where ε_M is the machine precision.

The perturbation E of the matrix A is in general not known; it may be due to errors in the underlying model, or due to data errors, so that the elements e_{ij} have some statistical distribution (for details see [20] and [105]), or the error can be due to the numerical computations. The value for the tolerance ε should then be chosen appropriately: if the error is due to round-off we can choose $\varepsilon \simeq \varepsilon_M \|A\|_\infty$; if the error in the model or in the data is larger than the round-off, then choose $\varepsilon \simeq 10^{-p}\|A\|_\infty$, when the elements of A have p correct decimal digits. It is important to stress that all the previous choices are only valid if the errors in all the elements of A are roughly of the same size, see [230].

Golub and Van Loan [105], p. 262, give two options to estimate the numerical rank if there is no clear gap in the singular values, both based on a minimization process: one option is when an accurate solution of the LSQ problem is important, and the other is used when minimizing the size of the residual is more important.

Example 65. *Consider the $n \times n$ matrix*

$$\begin{pmatrix} 0.1 & 1 & & & & \\ & 0.1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & & 0.1 & 1 \\ & & & & & 0.1 \end{pmatrix}.$$

The singular values of A in the case $n = 6$ are

$$\sigma_1 = 1.088, \quad \sigma_2 = 1.055, \quad \sigma_3 = 1.007,$$

$$\sigma_4 = 0.955, \quad \sigma_5 = 0.915, \quad \sigma_6 = 9.9 \cdot 10^{-7},$$

i.e., the numerical rank with respect to any threshold ϵ between 10^{-1} and 10^{-6} is $r_\epsilon(A) = n - 1$, whereas the mathematical rank is n. There is a distinct gap in the singular values.

5.2 Peters-Wilkinson LU factorization

The LU factorization method described in Section 4.2, can be modified to compute the pseudoinverse of a rank-deficient matrix $A \in \mathbb{R}^{m\times n}$ with $\text{rank}(A) = r < \min(m, n)$ and from it the minimum-norm solution of the least squares problem can be calculated. We recommend that this method

be used with caution, since rank determination in an LU factorization can be risky because the singular values of A change during the elimination process; see [162] for recent results on rank-revealing LU factorizations.

The algorithm uses Gaussian elimination with complete pivoting including linear independency checks. In the present case, the process will stop when there is no pivot larger than a specified tolerance. The factorization is then $\Pi_1 A \Pi_2 = LDU$. Now, both L and U have trapezoidal form of width r,

$$L = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix}, \qquad U = (\, U_1 \,,\, U_2 \,),$$

where L_1 and U_1 are nonsingular $r \times r$ matrices, lower and upper triangular, respectively, with unit diagonal, and D is diagonal as in Section 4.2. Complete pivoting ensures that L and U will be well conditioned and that D reflects $\mathrm{cond}(A)$.

A more convenient form of $\Pi_1 A \Pi_2$ is

$$\Pi_1 A \Pi_2 = \begin{pmatrix} I_r \\ T \end{pmatrix} L_1 D U_1 (\, I_r \,,\, S \,),$$

where $T = L_2 L_1^{-1}$ and $S = U_1^{-1} U_2$ are obtained using back-substitution on $T L_1 = L_2$ and $U_1 S = U_2$. The minimum-norm solution is then:

$$\widehat{\boldsymbol{x}}^* = A^\dagger \boldsymbol{b} = \Pi_2 (\, I_r \,,\, S \,)^\dagger U_1^{-1} D^{-1} L_1^{-1} \begin{pmatrix} I_r \\ T \end{pmatrix}^\dagger \Pi_1 \boldsymbol{b}.$$

In [22] there is also a modification of the Peters-Wilkinson algorithm for sparse problems. The package LUSOL [90] implements sparse LDU factorization with options for complete and rook pivoting, which also has rank-revealing properties while giving sparse factors. Rook pivoting is a strategy intermediate between full and row pivoting. At each step of elimination the pivot is chosen as one that is the largest in the row and column to which it belongs. The experimental performance of rook pivoting makes it comparable to full pivoting in precision and to partial pivoting in computational efficiency.

5.3 QR factorization with column permutations

The methods described in this section are particularly convenient for subset selection when a basic solution is wanted. Given A and $\boldsymbol{b}$ we want to find a permutation Π such that the first $r = \mathrm{rank}(A)$ columns of $A\Pi$ are linearly independent and also find a vector $\boldsymbol{z} \in \mathbb{R}^r$ that solves

$$\min_{\boldsymbol{z}} \| \boldsymbol{b} - A_r \boldsymbol{z} \|_2 \,, \; A_r = A \Pi \begin{pmatrix} I_r \\ 0 \end{pmatrix}.$$

For a QR factorization with column permutations, orthogonal transformations such as Householder reflections are applied to zero out successive sub-columns, interlaced with appropriate column permutations, to place the most linearly independent columns in front. At step k (after the transformation), the matrix has the form

$$R^{(k)} = Q_k^T A \Pi_k = \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & A_{22}^{(k)} \end{pmatrix} \begin{matrix} k \\ m-k \end{matrix},$$

with $Q_k^T = H_k \cdots H_1$ and where Π_k is the product of the permutation matrices used so far. In theory, one should be able to obtain an upper trapezoidal structured matrix after $k = \text{rank}(A)$ steps, while in practice – due to finite precision arithmetic – we must expect that $A_{22}^{(k)} \neq 0$ for all k. At each step k there are two important questions:

1. How to select the most linearly independent columns of $A_{22}^{(k)}$, i.e., what permutations should be performed.
2. How to decide when to stop the process because the numerical rank has been attained and $R_{11}^{(k)}$ contains the linearly independent information of A.

To formalize the second point we introduce the concept of a *rank-revealing QR* (RRQR) factorization, following [40]. Intuitively this is the QR factorization of a permutation of A chosen to yield a "small" $A_{22}^{(r)}$, where r is the numerical rank.

Definition 66. ***Rank-revealing QR (RRQR) factorization.*** *Let $A \in \mathbb{R}^{m\times n}$ have singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$, with a well-defined numerical rank r, i.e., with a clear gap between σ_r and σ_{r+1}. The factorization*

$$A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \begin{matrix} r \\ m-r \end{matrix}, \tag{5.3.1}$$

with R_{11} upper triangular of order r, is an RRQR factorization of A if

$$\text{cond}(R_{11}) \simeq \frac{\sigma_1}{\sigma_r} \qquad \textit{and} \qquad \|R_{22}\| \simeq \sigma_{r+1}.$$

This means that R_{11} contains the linearly independent information of A and R_{22} is a submatrix of small norm, in which case the matrix

$$W = \Pi \begin{pmatrix} R_{11}^{-1} R_{12} \\ -I \end{pmatrix} \tag{5.3.2}$$

is a good first approximation for the basis of the null space of A. The null vectors can be improved by a few steps of subspace iteration as shown in

[42]. If after k steps the above two conditions of RRQR hold with $r = k$, the best possible factorization has been attained and the numerical rank is determined as r.

Hong and Pan [129] proved that there always exists a permutation matrix Π so that $A\Pi = QR$ is an RRQR factorization of A if the numerical rank is known. The question is how to obtain an economic and reliable algorithm. One can reformulate the RRQR conditions in terms of a bi-objective optimization processes, i.e., the task is to find a permutation Π that maximizes $\sigma_r(R_{11})$, the smallest singular values of R_{11} and at the same time minimizes $\sigma_1(R_{22})$, the norm of R_{22}.

The maximization ensures that the first r columns of Q span the range of A. The *pivoted QR factorization*, a column pivoting strategy devised by Businger and Golub (see [105], p. 235), fulfills this requirement. At step k, the column of the rectangular matrix $A_{22}^{(k)}$ with largest norm is permuted to be in the first position, after which the Householder transformation that zeros all the elements of this sub-vector below the diagonal is applied. This algorithm is implemented in Linpack and Lapack and can be applied successfully in most cases if it is used with some safeguarding condition estimation (see, for example, [20], pp. 114ff.).

Theoretically, this process should be continued up to $k = \text{rank}(A)$. As the rank is unknown, the stopping criterion in the Businger-Golub algorithm is $\|A_{22}^{(k)}\|_2 \leq \varepsilon_M \|A\|_2$. This criterion works well in most cases, but may fail occasionally because, although a sufficiently small $A_{22}^{(k)}$ means that A is close to a matrix of rank k, it may happen that $R_{11}^{(k)}$ is nearly rank deficient without $A_{22}^{(k)}$ being very small and the algorithm should have been stopped.

Example 67. *The following pathological example due to Kahan (see, for example, [20]) illustrates the fact that the column pivoted QR factorization algorithm can fail. Let*

$$A = \text{diag}(1,\, s,\, \ldots,\, s^{n-1}) \begin{pmatrix} 1 & -c & -c & \cdots & -c \\ & 1 & -c & \cdots & -c \\ & & \ddots & & \\ & & & \ddots & -c \\ & & & & 1 \end{pmatrix}. \tag{5.3.3}$$

Assume that $n = 100$, $c = 0.2$ and $s = 0.9798$. Then A is invariant under the previous algorithm and $A_{22}^{(n-1)} = a_{nn} = 0.133$; however, A is almost rank deficient with $\text{rank}(A) = n - 1$, as can be seen from the singular values $\sigma_{n-1} = 0.1482$ and $\sigma_n = 0.368 \cdot 10^{-8}$. It was proved in [262] that $\sigma_{n-1} = s^{n-2}\sqrt{1+c}$.

Several hybrid algorithms that solve the second minimization have been designed. They use the Businger-Golub algorithm in a pre-processing stage and then they compute another permutation matrix to improve on the rank-revealing aspect. The theoretical basis for most of these algorithms is the following bound from [42], on the norm of the lower right sub-block in the RRQR factorization:

$$\|R_{22}\|_2 \leq \|A\,\Pi\,Y\|_2 \left\|Y_2^{-1}\right\|_2, \qquad Y = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} \begin{matrix} r \times (n-r) \\ (n-r) \times (n-r) \end{matrix},$$

where the linearly independent columns of $\Pi\,Y$ are approximate null vectors of A. The norm of R_{22} will be small if one chooses a Y and a permutation Π such that Y_2 is well conditioned.

Assume, for example, that $\mathrm{rank}(A) = n-1$, that the pivoting algorithm has been applied and a preliminary QR factorization has been obtained with R_1 triangular. The null space of A is spanned by $\boldsymbol{v}_n$, the right singular vector corresponding to σ_n. An approximating vector $\boldsymbol{y} \simeq \boldsymbol{v}_n$ can be obtained by a couple of inverse iteration steps on $R_1^T R_1$ (see the subsection on properties of the SVD). The permutation Π is then determined by moving the largest component in absolute value of $\boldsymbol{y}$ to the end, so that $\|Y_2\|_2 = |v_n|$ is large. This technique is then adapted to the general case when $\mathrm{rank}(A) < n-1$, see [40] for details.

An efficient RRQR algorithm was developed by refining the pivoted QR factorization (see [14] for a survey and description). The algorithm can be found in [263] under TOMS/782, and it starts with a pre-processing using pivoted QR, followed by interlaced column permutations with re-triangularization to obtain the rank-revealing condition. A more expensive algorithm to determine the $r < n$ most linearly independent columns of A is an SVD-based algorithm developed by Golub, Klema and Stewart [97]. It combines subset selection using SVD with the computation of the basic solution using QR factorization (see [20], pp. 113, for details).

Once the RRQR factorization (5.3.1) has been computed, the basic solution $\boldsymbol{x}_{\mathrm{B}}$ is easily computed by multiplying $\boldsymbol{b}$ with Q_1^T, followed by back-substitution with R_{11}, which costs a total of $4mn - 2n^2 + r^2$ flops. To compute the LSQ solution of minimal norm (following [20], p. 107), we must first compute the matrix W in (5.3.2) and then we have

$$\widehat{\boldsymbol{x}}^* = \boldsymbol{x}_{\mathrm{B}} - W\,\boldsymbol{x}_{\mathrm{N}}, \qquad \boldsymbol{x}_{\mathrm{N}} = W^{\dagger}\boldsymbol{x}_{\mathrm{B}},$$

in which $\boldsymbol{x}_{\mathrm{N}}$ is computed by solving the least squares problem $\min_{\boldsymbol{z}} \|W\,\boldsymbol{z} - \boldsymbol{x}_{\mathrm{B}}\|_2$ (we note that the vector $W\,\boldsymbol{x}_{\mathrm{N}} = W\,W^{\dagger}\boldsymbol{x}_{\mathrm{B}}$ is the component of $\boldsymbol{x}_{\mathrm{B}}$ in the null space of A that we subtract from $\boldsymbol{x}_{\mathrm{B}}$ to arrive at the minimal-norm solution). The total cost of this approach is $r^2(n-r) + 2r(n-r)^2 + 2r^2$ flops.

5.4 UTV and VSV decompositions

A *UTV decomposition* is a slightly different type of decomposition, where the matrix A is written as the product of three matrices, two orthogonal ones and a middle upper or lower triangular (or block triangular) matrix. These decompositions fill a space somewhere between the pivoted QR factorization and the SVD. They provide orthonormal bases for the range and null space, as well as estimates for the singular values – and they are faster to compute and update. The algorithms to compute a URV decomposition start with a QR step, followed by a rank revealing stage when the singular vectors corresponding to the smaller singular values are estimated.

The URV decomposition, with an upper triangular middle matrix, takes the following form:

Definition 68. *Given the $m \times n$ matrix A, a URV decomposition is a factorization of the form*

$$A = U_R \begin{pmatrix} R \\ 0 \end{pmatrix} V_R^T = \begin{pmatrix} U & U_{R_o} & U \end{pmatrix} \begin{pmatrix} R_k & F \\ 0 & G \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_{R_k} & V \end{pmatrix},$$

where R_k is a $k \times k$ nonsingular upper triangular matrix and G is $(n-k) \times (n-k)$. If A has a well-defined gap between the singular values, $\sigma_{k+1} \ll \sigma_k$, then the URV decomposition is said to be rank revealing if

$$\sigma_{\min}(R_k) = O(\sigma_k) \qquad \text{and} \qquad \|(F\ G)\|_2 = O(\sigma_{k+1}).$$

There is also a ULV decomposition, where the matrix L is lower triangular – we skip the details here and refer instead to [78]. The QR factorization with column permutations can be considered as a special URV decomposition, and the justification is the following theorem [129].

Theorem 69. *For $0 < k < n$ define*

$$c_k = \sqrt{k(n-k) + \min(k, (n-k))}.$$

Then there exists a permutation matrix Π such that the pivoted QR factorization has the form

$$A\,\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix},$$

with a $k \times k$ upper triangular R_{11}, $\sigma_k(R_{11}) \geq \sigma_k(A)$ and $\|R_{22}\|_2 \leq c_k\, \sigma_{k+1}(A)$.

The normal equations can also handle rank-deficient problems, as long as the numerical rank of A is well defined. The starting point is a Cholesky factorization of $A^T A$ using symmetric pivoting,

$$\Pi^T (A^T A)\, \Pi = R_1^T R_1,$$

where Π is a permutation matrix, and R_1 is the upper triangular (or trapezoidal) Cholesky factor; the numerical properties of this algorithm are discussed by Higham [127]. The next step is to compute a ULV decomposition of the matrix $E\,R_1 E$, where the "exchange matrix" E consists of the columns of the identity matrix in reverse order,

$$E\,R_1 E = U_L\,L\,V_L^T, \qquad L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix},$$

which leads to the symmetric *VSV decomposition* of $A^T A$, having the form

$$A^T A = V\,L^T L\,V^T, \qquad V = \begin{pmatrix} V_1 & V_2 \end{pmatrix} = \Pi\,E\,V_L$$

(the orthogonal factor U_L is not used). Then the minimal norm solution is given by

$$\widehat{\boldsymbol{x}}^* = V_1 L_{11}^{-1} L_{11}^{-T} V_1^T (A^T b).$$

We refer to [124] for more details about definitions and algorithms for symmetric VSV decompositions.

5.5 Bidiagonalization

We now describe a procedure based on Householder transformations of the compound matrix $\begin{pmatrix} \boldsymbol{b} & A \end{pmatrix}$ for computing a matrix bidiagonalization (see [21] for more details). The process is also used in the solution of total least squares problems. The same least squares problem structure can also be achieved with a Lanczos-type process, as will be defined for use in the LSQR algorithm in the next chapter, although the present method is more stable.

For simplicity, we will assume first that A has full rank. The objective is to use Householder transformations to compute orthogonal matrices $U_B \in \mathbb{R}^{m\times m}$ and $V_B \in \mathbb{R}^{n\times n}$, such that $U_B^T A V_B$ is lower bidiagonal. For convenience, instead of bidiagonalizing A and then computing $U_B^T \boldsymbol{b}$, we will apply the Householder transformations to the augmented matrix $\begin{pmatrix} \boldsymbol{b} & A \end{pmatrix}$. This has the advantage that the transformed problem has a very simple right-hand side. We use a sequence of interleaved left and right Householder reflections, so that in step $k = 1, \ldots, n+1$ a left reflection zeros the elements in column k under the diagonal of the augmented matrix and a right reflection zeros the elements in row k from element $(k, k+1)$ onward

of A, resulting in the decomposition

$$U_B^T(\ \boldsymbol{b}\quad AV_B\ ) = \begin{pmatrix} \beta_1 & \alpha_1 & & & \\ 0 & \beta_2 & \alpha_2 & & \\ & & \ddots & \ddots & \\ & & & \beta_n & \alpha_n \\ & & & & \beta_{n+1} \\ 0 & & \cdots & & 0 \\ \vdots & & & & \vdots \end{pmatrix} \quad \text{with} \quad \beta = \|\boldsymbol{b}\|_2.$$

With $\boldsymbol{x} = V_B\boldsymbol{y}$ we now obtain

$$\begin{aligned} &\min_{\boldsymbol{x}} \|A\,\boldsymbol{x} - \boldsymbol{b}\|_2 = \min_{\boldsymbol{y}} \|U_B^T A V_B \boldsymbol{y} - U_B^T \boldsymbol{b}\|_2 = \\ &\min_{\boldsymbol{y}} \|B\boldsymbol{y} - \beta_1 \widehat{\boldsymbol{e}}_1\|_2, \end{aligned} \tag{5.5.1}$$

with the $(n+1) \times n$ matrix

$$B = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_n & \alpha_n \\ & & & \beta_{n+1} \end{pmatrix}.$$

After $k < n$ steps of the bidiagonal reduction one has computed orthogonal matrices U_{k+1} and V_k such that their columns are the first $k+1$ and k columns in U_B and V_B respectively. $U_k^T A V_k = B_k$ is the leading $(k+1) \times k$ submatrix of B.

Note that for the above-described bidiagonalization, as before for the QR factorization method, no significant additional storage is needed because the relevant information about the transformations can be stored in the empty spaces of A. The complete bidiagonal factorization applied only to A can be used as a first step of an algorithm for calculating the SVD – see the next section.

Another bidiagonalization algorithm, which is advantageous in the case that $m \gg n$, was first described in [150], and it is known as R-bidiagonalization. Instead of bidiagonalizing the original matrix A, the algorithm starts by a QR factorization and then it bidiagonalizes the resulting upper triangular matrix. In the second step the algorithm can take advantage of the special structure and uses the less expensive Givens transformations. The number of flops is changed from $4mn^2 - 4n^3/3$ for the standard bidiagonalization to $2mn^2 + 2n^3$ for R-bidiagonalization, so the break-even point is $m = 5n/3$.

The bidiagonalization can be stopped prematurely if a zero element appears in B. One obtains then a so-called core least squares problem, with important properties that simplify the solution of the LS problem; see [180]. In fact, if $\alpha_{k+1} = 0$, the least squares problem takes the form

$$\min_y \left\| \begin{pmatrix} B_k & 0 \\ 0 & A_k \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix} - \begin{pmatrix} \beta_1 \widehat{\boldsymbol{e}}_1 \\ 0 \end{pmatrix} \right\|_2 .$$

Thus, it can be decomposed into two independent minimization problems:

$$\min_{\boldsymbol{y}_1} \|B_k \boldsymbol{y}_1 - \beta_1 \widehat{\boldsymbol{e}}_1\|_2 , \quad \min_{\boldsymbol{y}_2} \|A_k \boldsymbol{y}_2\|_2 .$$

The first is the core subproblem; it has a full-rank matrix and hence a unique solution that can be computed as above via QR with Givens rotations. The minimum of the second problem is obtained for $\boldsymbol{y}_2 = 0$. A similar argument can be used to decompose the problem (5.5.1) into two separable ones, if an element $\beta_{k+1} = 0$.

Using the fact that at each step the bidiagonal reduction produces B_k, the leading submatrix of the final B, Paige and Saunders derived a technique called LSQR to solve general sparse rank-deficient least squares problems [179]. Essentially, it interleaves QR solution steps with an iterative formulation of the reduction to bidiagonal form in order to compute a sequence of solutions $\boldsymbol{y}_k \in \mathbb{R}^k$ to the reduced problem

$$\min_{\boldsymbol{y}} \|\beta_1 \widehat{\boldsymbol{e}}_1 - B_k \boldsymbol{y}\|_2 .$$

By applying Givens rotations, this lower bidiagonal problem can be transformed to an equivalent upper bidiagonal one, which in the LSQR case has specific computational advantages: the sequence of residual norms is decreasing. One accepts $\boldsymbol{x}_k = V_k \boldsymbol{y}_k$ as an approximate solution of the original least squares problem if $\frac{\|A^T \boldsymbol{r}_k\|_2}{\|A\|_2 \|\boldsymbol{r}_k\|_2}$ is smaller than a given tolerance. The method is also known as partial least squares (PLS); see, for example, [256], [257].

5.6 SVD computations

We conclude this chapter with a brief discussion of the Golub-Reinsch-Kahan algorithm for computing the SVD; details can be found in [20] and [105]. The steps are

1. Bidiagonalization of A by a finite number of Householder transformations from left and right,

$$U_B^T A \, V_B = \begin{pmatrix} B \\ 0 \end{pmatrix}, \qquad B \text{ bidiagonal.}$$

This is the closest to a diagonal form achievable by a finite process.

	SVD (G-R-K)	R-SVD	To obtain
Σ	$4mn^2 - 4n^3/3$	$2mn^2 + 2n^3$	rank
Σ, V, $U_n^T \boldsymbol{b}$	$4mn^2 + 8n^3$	$2mn^2 + 11n^3$	LSQ solution
Σ, V, U_n	$14mn^2 + 8n^3$	$6mn^2 + 20n^3$	pseudoinverse

Table 5.1: Comparison of algorithms for computing the SVD: the standard Golub-Reinsch-Kahan algorithm and the variant with an initial QR factorization. The matrix U_n consists of the first n columns of U.

Method	Used for	Cost
LU (Cholesky)	min solution	$2mn^2 - \frac{2}{3}n^3$
QR Householder	factorization	$4mnr - 2r^2(m+n) + \frac{4}{3}r^3$
QR Householder	basic solution	$4mn - 2n^2 + r^2$
QR Householder	min solution	$r^2(n-r) + 2r(n-r)^2 + 2r^2$
Bidiagonalization	min solution	$4mn^2 - \frac{4}{3}n^3$
R-bidiagonalization	min solution	$2mn^2 + 2n^3$
SVD (G-R-K)	min solution	$4mn^2 + 8n^3$
R-SVD	min solution	$2mn^2 + 11n^3$

Table 5.2: Comparison of some methods to solve the LSQ problem in the rank-deficient case, where r is the rank of A. The LU and QR factorizations use permutations.

2. Computation of the SVD of B by an iterative procedure, which produces:
$$U_\Sigma^T B V_\Sigma = \Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n).$$

3. Finally, the SVD of A is $A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T$, with
$$U = U_B \begin{pmatrix} U_\Sigma & 0 \\ 0 & I_{m-n} \end{pmatrix} \qquad \text{and} \qquad V = V_B V_\Sigma.$$

For the second step, an algorithm that is equivalent to an implicit-shift QR for $B^T B$ is applied directly to B in order to zero its superdiagonal elements. Basically the procedure consists of Givens rotations, alternately applied to the right and left that chase down the nonzero off-diagonal elements. The equivalence with the implicit-shift QR guarantees the convergence, which is often cubic. As mentioned in the previous section, this algorithm can be extended with a pre-processing stage that computes a QR factorization of A, applies steps 1–3 to R, and finally updates U by replacing it for $Q\,U$.

Table 5.1, with information taken from [105], compares the properties and costs of the SVD algorithms with and without R-bidiagonalization.

The matrices Σ and U_n correspond to the economical version of the SVD. For the LSQ problem, the matrix U need not be explicitly formed because it can be applied to $\boldsymbol{b}$ as it is developed. The numbers above assume that the iterative stage 2 needs on average 2 steps per singular value. If A is nearly rank deficient, this will be reflected in the SVD, and the relative accuracy of the smaller singular values will suffer.

A variant of the Golub-Reinsch-Kahan SVD algorithm was described in [248]; this algorithm is called partial SVD, and it terminates the iterations in step 2, once it is guaranteed that all remaining singular values to be computed are smaller than a user-specified threshold.

Table 5.2, using information from [105], compares some of the methods discussed in this chapter for matrices of rank $r < \min(n, m)$.

This page intentionally left blank

Chapter 6

Methods for Large-Scale Problems

Large-scale problems are those where the number of variables is such that direct methods, like the ones we have described in earlier chapters, cannot be applied because of storage limitations, accuracy restrictions or computational cost. Usually, large-scale problems have special structure, such as sparseness, which favors the use of special techniques. We already saw that Givens rotations are preferable to Householder transformations in this case.

There are many iterative methods tailored to specific problems and applications in this area; here we choose to survey only the most important algorithms and concepts related to iterative methods for linear LSQ problems. We finally discuss a block approach that can be combined with either direct or iterative methods and can handle fairly large problems in a distributed network of computers. The reader is referred to [20] for a general survey of methods.

6.1 Iterative versus direct methods

If the LSQ problem is large, the question arises whether either iterative or direct methods should be used. A problem with, for example, half a million equations and $n = O(10^5)$ parameters to be determined, such as is common in solving inverse problems, needs conservatively $O(n^3)$ operations if one uses a direct method. On a modern computer (circa 2011), performing 10^{10} floating-point operations/second, that calculation would require about 28 hours of CPU time, although hardware is getting cheaper and faster all the time. Codes that take advantage of multicore hardware can also speed up the process.

On the other hand, iterative methods compute an approximation to the solution and then improve on it until the error is small enough. For many large problems, iterative methods are favorable alternatives to direct methods because they can (approximately) solve the problem faster and with much less demands on memory. Typically, iterative methods do not require the matrix to be explicitly stored – instead they use subroutines that compute the necessary matrix-vector multiplications.

However, if the normal equations are sparse, a direct method may still be a good option. The advantages/disadvantages of direct and iterative methods can be summarized as follows:

- **Speed**

  Direct methods: the number of necessary steps is known, but fill-in (i.e., creation of new nonzero elements) may make the process prohibitively expensive. Taking advantage of sparseness to minimize fill-in is an art and it requires special algorithms.

  Iterative methods: at best there is an estimate of the number of iterations and the cost per iteration, but matrix structures are preserved.

- **Storage**

  Direct methods: sometimes efficient compressed storage formats can be used but retrieval may be costly.

  Iterative methods: compressed storage formats can be chosen, subject to the same proviso as above.

- **Robustness**

  Direct methods: there is a direct method applicable to every problem and the solution can always be attained.

  Iterative methods: may require many iterations to converge.

- **Formulation**

  Direct methods: solve the normal equations (this may destroy sparseness) or the overdetermined system itself.

  Iterative methods: solve the normal equations $A^T(A\boldsymbol{x}-\boldsymbol{b})$, the overdetermined system, or the augmented matrix form.

A compromise between the two approaches are preconditioned iterative methods, which use an approximate factorization of A. Also, block iterative methods that are easily parallelizable are an interesting alternative discussed at the end of the chapter.

For the sake of completeness, the classical iterative methods will be surveyed first, and then some of the more efficient Krylov subspace methods will be explained.

6.2 Classical stationary methods

The basic idea behind the definition of the classical iterative methods is to split the matrix so that the resulting system can easily be solved at each iteration. We consider first the application to the normal equations. Recall that if A is a full-rank matrix, then the normal equations are symmetric and positive definite. The general formulation is given below:

- Split $A^T A$ into convenient matrices $Q-(Q-A^T A)$, where Q is easily invertible.
- The iteration is now defined by choosing a starting vector $\boldsymbol{x}^{(0)}$ and then solving at each step: $Q\,\boldsymbol{x}^{(k+1)} = (Q - A^T A)\,\boldsymbol{x}^{(k)} + A^T \boldsymbol{b}$.
- The convergence properties depend on the spectral radius of the iteration matrix: $I - Q^{-1}(A^T A)$.

The most common methods are

- Richardson: $Q = I$.
- Jacobi: $Q =$ main diagonal of $A^T A$.
- Gauss-Seidel: $Q =$ lower triangular part of $A^T A$. It can be shown that this is a residual-reducing method.
- SOR: introduces a relaxation factor ω in the Gauss-Seidel method. Convergence is assured if the acceleration parameter satisfies $\omega \in (0\,,\,2)$.

We point out that all these methods can be implemented without forming the product $A^T A$ and that their convergence in the full-rank case is well known. There are also convergence results for the rank-deficient case. A not so well known fact is that splitting can also be applied directly to the rectangular matrix A, thus avoiding the use of the normal equations and its poorer conditioning. Not every splitting is valid in this case and one needs to use a so-called proper splitting.

Definition 70. *Given the matrix A, the splitting $A = Q - (Q - A)$ is a proper splitting if the ranges and null spaces of A and Q coincide.*

For a proper splitting, the iteration $Q\,\boldsymbol{x}^{(k+1)} = (Q - A)\,\boldsymbol{x}^{(k)} + \boldsymbol{b}$ (the problem to solve at each step is a least squares problem) converges to $\boldsymbol{x}^* = A^{\dagger}\boldsymbol{b}$ if and only if the spectral radius ρ of the iteration matrix satisfies $\rho(Q^{\dagger}(Q - A)) < 1$. For details see [20].

If the rate of convergence for an iterative method is slow, there are several possible ways to improve it, such as Chebyshev acceleration or preconditioning. In the Chebyshev method, k original iterates are combined

to define an iteration matrix with a smaller spectral radius. The other convergence acceleration method, preconditioning, is also used extensively for Krylov methods and will therefore be explained after these methods have been introduced.

6.3 Non-stationary methods, Krylov methods

In contrast to the classical iterative methods described above, Krylov methods have no iteration matrix. Their popularity, both for the solution of systems of equations and for least squares problems, is because, in addition to low computing time and memory requirements, they are parameter free. Access to A is only via subroutines for matrix/vector products, and therefore they are well suited to solving large, sparse problems. The convergence rate for well-conditioned matrices is good, usually reaching an acceptable accuracy in a small number of steps. In addition, the rate of convergence can be improved with various preconditioning techniques. First we define Krylov subspaces.

Definition 71. *Given an $m \times m$ matrix A, a vector $\boldsymbol{z}$ and an integer d, the Krylov subspace denoted by $K_d(A, \boldsymbol{z})$ is the space generated by*

$$K_d(A, \boldsymbol{z}) \equiv \text{span}\left\{\boldsymbol{z}, A\boldsymbol{z}, \ldots, A^{d-1}\boldsymbol{z}\right\}.$$

Solution of square systems

In this subsection we consider the linear system $A\,\boldsymbol{x} = \boldsymbol{z}$, where A denotes a square matrix. Krylov subspace methods for the solution of nonsingular linear systems $A\,\boldsymbol{x} = \boldsymbol{z}$ are based on the expansion of A^{-1} in powers of the matrix A using its minimal polynomial

$$q(t) = \prod_{j=1}^{d} (t - \lambda_j)^{m_j},$$

where λ_j are the distinct eigenvalues of A with corresponding multiplicity m_j. The inverse of the matrix can then be written as

$$A^{-1} = \sum_{i=0}^{m-1} \gamma_i A^i,$$

with some coefficients γ_i and $m = \sum_{j=1}^{d} m_j$. The methods are also applicable sometimes to the singular matrix case, based on the Drazin inverse solution. For a very clear introduction see [133].

Given the power expansion of A^{-1}, it is easy to see that the linear system solution: $\boldsymbol{x} = A^{-1}\boldsymbol{z}$, belongs to a Krylov subspace generated by

A and $\boldsymbol{z}$. The dimension d of this space is defined by the degree of the minimal polynomial of A and therefore by its eigenvalue distribution. It will be smaller if there are clusters of eigenvalues (especially if their associated eigenvectors are linearly independent). Faster convergence of Krylov algorithms is related to smaller dimensions.

Krylov algorithms compute successive approximations $\boldsymbol{x}^{(k)}$ to the solution in expanding Krylov spaces $K_k(A, \boldsymbol{z})$ for $k = 1, \ldots, d$. Some of the methods recast the approximations in the form: $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(0)} + \boldsymbol{p}^{(k)}$, with $\boldsymbol{x}^{(0)}$ the initial guess and $\boldsymbol{p}^{(k)}$ *direction* vectors. This can be viewed as solving the problem: $A\,\boldsymbol{p} = \boldsymbol{z} - A\boldsymbol{x}^{(0)} \equiv \boldsymbol{r}^{(0)}$, with the $\boldsymbol{p}^{(k)}$ approximations to $\boldsymbol{p}$ belonging to the Krylov space $K_k(A, \boldsymbol{r}^{(0)})$.

Conjugate gradient method for LSQ problems

The conjugate gradient (CG) method (see the original Hestenes and Stiefel paper [126]), applicable to a symmetric positive definite system $A\boldsymbol{x} = \boldsymbol{z}$, is a Krylov subspace method that at the kth iteration minimizes the functional

$$\Phi(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\,\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{z}$$

over the affine subspace $\boldsymbol{x}^{(0)} + K_k(A, \boldsymbol{r}^{(0)})$. Note that a minimum of $\Phi(x)$ occurs when $A\,\boldsymbol{x} = \boldsymbol{z}$.

If A has full column rank so that A^TA is positive definite, CG can be applied to the normal equations in factored form. An implementation can be found in the program CGLS; see [263]. The algorithm generates solution approximations of the form $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} + \boldsymbol{p}^{(k-1)}$.

The solution of the problem $A^TA\,\boldsymbol{p} = A^T(\boldsymbol{z} - A\,\boldsymbol{x}^{(0)}) \equiv A^T\boldsymbol{r}^{(0)}$ is achieved by minimizing the error functional in the A^TA-norm,

$$\|\boldsymbol{x}^* - \boldsymbol{x}\|^2_{A^TA} \equiv (\boldsymbol{x}^* - \boldsymbol{x})^T A^T A\,(\boldsymbol{x}^* - \boldsymbol{x}),$$

over $\boldsymbol{x}^{(0)} + K_k(A^TA, A^T\boldsymbol{r}^{(0)})$, by moving along a set of A^TA-orthogonal direction vectors $\boldsymbol{p}^{(k)}$.

At every iteration an update $\boldsymbol{r}^{(k)}$ of the residual and a new direction vector $\boldsymbol{p}^{(k)}$ is calculated, so that the vectors $\{\boldsymbol{p}^{(k)}\}_{k=1,\ldots}$ form an A^TA-orthogonal set (or conjugate orthogonal), i.e., $(\boldsymbol{p}^{(k)})^TA^TA\,\boldsymbol{p}^{(j)} = 0$, $j = 1, \ldots, k-1$ and the residuals $\{\boldsymbol{r}^{(j)}\}$, $j = 1, \ldots, k$ form an orthogonal set.

Algorithm CGLS

Initialize: $\boldsymbol{x}^{(0)} = 0$, $\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$, $\boldsymbol{p}^{(0)} = A^T\boldsymbol{r}^{(0)}$

for $k = 1, 2, \ldots$

Compute the step by minimizing $\Phi(x)$ along the direction $\boldsymbol{p}^{(k-1)}$

```
    Update x^(k) and r^(k)
    Compute the new direction p^(k), A^T A-orthogonal
    to all previous directions
    Check for convergence
end
```

Only two coupled two-term recurrences are needed: one to update the residuals and another to update the directions, involving a total of $3n+2m$ flops/step, in addition to two matrix-vector multiplications (with A and A^T) per step. The iteration vectors $\boldsymbol{x}^{(k)}$ converge to the LSQ solution $\boldsymbol{x}^*$ if $\boldsymbol{x}^{(0)} \in \text{range}(A^T)$, a condition that holds if the initial vector is chosen as $\boldsymbol{x}^{(0)} = \boldsymbol{0}$. The rate of convergence can be estimated by

$$\left\|\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\right\|_{A^TA}^2 < 2\left(\frac{\text{cond}(A^TA)-1}{\text{cond}(A^TA)+1}\right)^k \left\|\boldsymbol{x}^* - \boldsymbol{x}^{(0)}\right\|_{A^TA}^2.$$

The convergence is very fast if $\text{cond}(A)^2 \sim 1$, i.e., when the columns of A are approximately orthogonal.

In exact arithmetic, the number of iterations to achieve convergence is equal to the number of distinct singular values of A. In general this number is not known, so an upper bound is the dimension n, but in practice it may need far fewer or far more iterations because of the influence of rounding errors on the orthogonalization process. Another important property is that the norms of the error and the residual decrease monotonically with k, but for ill-conditioned matrices A, the norm $\left\|A^T\boldsymbol{r}^{(k)}\right\|_2$ may oscillate as it converges to zero.

Example 72. ***Computed Tomography.*** *In medicine, materials science, geoscience and many other areas, one is interested in computing an image of an object's interior without opening or destroying the object. A well-known method for doing so is computed tomography (CT), where one records the damping of a multitude of X-rays (or other signals or waves) sent through the object from different directions. The damping of each ray or its time of travel is given by a line (or curvilinear) integral along the ray of the object's spatially varying absorption cofficient (or velocity of propagation) and an image of this absorption coefficient is then reconstructed.*

When the CT problem is discretized, we obtain a model in the form of an LSQ problem with a solution $\boldsymbol{x}^*$ whose elements represent the pixels in the image (or some other more concise parametrization of the medium). The right-hand side $\boldsymbol{b}$ consists of the measured dampings or travel times and the elements of A are related to the lengths of each of the rays through the individual pixels of the image. Since each ray only touches a small fraction of all the pixels in the image, the matrix A is very sparse, and we can use CGLS to solve the associated LSQ problem.

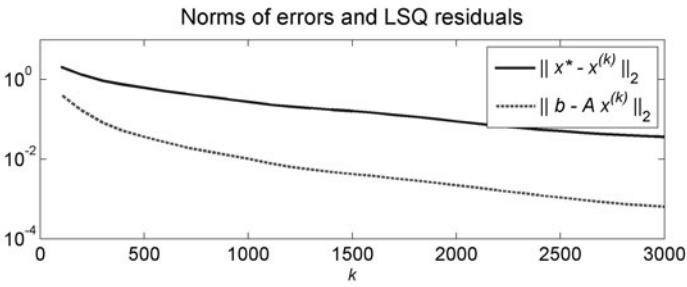

Figure 6.3.1: The norms of the errors and the LSQ residuals as a function of the number of iterations for the computed tomography problem.

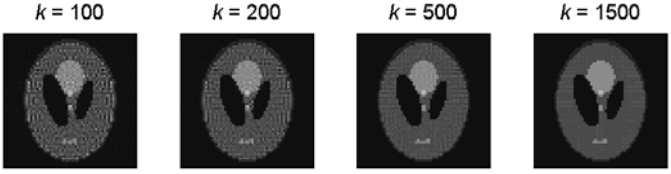

Figure 6.3.2: The CT reconstructions after 100, 200, 500 and 1500 CGLS iterations.

In this example with X-rays, we use a problem with 4096 *unknowns, corresponding to a* 64×64 *image and a total of* 4776 *rays, leading to a* 4776×4096 *sparse matrix* A *with a density of nonzero elements equal to 1.5% (the problem was generated with software from the AIR Tools package [123]). To simplify the example there is no noise. We performed 3000 CGLS iterations with this system. The norms of the errors* $\|\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\|_2$ *and the LSQ residuals* $\|A\,\boldsymbol{x}^{(k)} - \boldsymbol{b}\|_2$ *as functions of the iteration number* k *are shown in Figure 6.3.1. Clearly, both norms decay monotonically. Figure 6.3.2 shows the CT reconstructions after 100, 200, 500 and 1500 CGLS iterations, and we see that the amount of artifacts and inaccuracies is reduced considerably after 1500 iterations.*

For seismic tomography via ray tracing in elastic media see [187, 189].

Bidiagonalization method: LSQR

A mathematically equivalent Krylov process can be generated by a Golub-Kahan-type bidiagonalization method. An efficient implementation developed by Paige and Saunders in [179] is the LSQR code [263], which we describe now. It has been used to solve singular and ill-conditioned problems and has been shown by example to be numerically very reliable. Es-

sentially, it interleaves the reduction to bidiagonal form with partial solution steps in order to compute a sequence of approximating solutions $\boldsymbol{x}^{(k)} \in K_k(A^TA,\, A^T\boldsymbol{b})$, which are the minimizers of bidiagonal LSQ problems defined by a partial factorization of the matrix A.

In Section 5.5 we discussed an algorithm, based on interleaved left and right Householder transformations applied to the augmented matrix $(\,\boldsymbol{b}\;A)$, that results in a factorization of the matrix $A \in \mathbb{R}^{m\times n}$ into a product of two orthogonal matrices and a lower bidiagonal $m \times n$ matrix containing the nonsingular information of A:

$$A = U \begin{pmatrix} B \\ 0 \end{pmatrix} V^T,$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices with $U^T\boldsymbol{b} = (\beta_1, 0, \ldots, 0)^T$, $\beta_1 = \|\boldsymbol{b}\|_2$, and B is the lower bidiagonal matrix

$$B = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_n \\ & & & \beta_{n+1} \end{pmatrix} \in \mathbb{R}^{(n+1)\times n}.$$

The disadvantage of this *stable* algorithm is that the Householder transformations destroy the structure (in particular the sparsity) of A.

An alternative way to obtain the factorization is to generate the columns of the orthogonal matrices and the elements of B sequentially after equating the columns in the products $AV = U_1B$ and $A^TU_1 = VB^T$. Here $U_1 = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n+1})$ (see Lanczos methods in chapter 9 of [105]). The scalars α_j and β_j are determined so that $\|\boldsymbol{u}_j\|_2 = \|\boldsymbol{v}_j\|_2 = 1$, and $\boldsymbol{u}_1 = \boldsymbol{b}/\,\|\boldsymbol{b}\|_2$. Using the recursive formulas for $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$ one can show that $\boldsymbol{u}_j \in K_j(AA^T,\, \boldsymbol{u}_1)$ and $\boldsymbol{v}_j \in K_j(A^TA,\, A^T\boldsymbol{u}_1)$.

The sequence of approximate bidiagonal LSQ problems arises in the following way. At step k, the bidiagonalization process has produced three matrices: $U_{k+1} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{k+1})$, $V_k = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$ (submatrices of U and V) and B_k, the leading $(k+1) \times k$ submatrix of B.

An approximate solution $\boldsymbol{x}^{(k)}$ of the LSQ problem $\min_{\boldsymbol{x}} \|A\,\boldsymbol{x} - \boldsymbol{b}\|_2$, is now sought in the Krylov subspace $K_k(A^TA,\; A^T\boldsymbol{b})$ generated by the columns of $V_k = \mathrm{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\}$, i.e., we are looking for a linear combination of columns of V_k:

$$\boldsymbol{x}^{(k)} = V_k\boldsymbol{y}_k.$$

Using the recurrence relations, one obtains

$$\boldsymbol{b} - A\,\boldsymbol{x}^{(k)} = U_{k+1}(\beta_1\widehat{\boldsymbol{e}}_1 - B_k\boldsymbol{y}_k),$$

where $\widehat{\boldsymbol{e}}_1 = (1, 0, \ldots, 0)^T$ is the first unit vector and $\beta_1 = \|\boldsymbol{b}\|_2$. Because U_{k+1} is theoretically orthonormal, the LSQ problem in $\boldsymbol{x}^{(k)}$ can be replaced by a very simple one in $\boldsymbol{y}_k$:

$$\min_{\boldsymbol{x}^{(k)} \in \text{range}(V_k)} \left\|\boldsymbol{b} - A\,\boldsymbol{x}^{(k)}\right\|_2 = \min_{\boldsymbol{y}_k} \|\beta_1 \widehat{\boldsymbol{e}}_1 - B_k \boldsymbol{y}_k\|_2 .$$

This kth subproblem has a bidiagonal matrix and a special right-hand side given by $(\beta_1, 0, \ldots, 0)^T$. It can be solved with a QR factorization of B_k, using Givens rotations to take advantage of the bidiagonal structure. The computations can be organized in an efficient way so that at each step the factorization is developed from the previous one, adding only the necessary rotations to transform the additional column. Similarly, the iteration vector $\boldsymbol{x}^{(k)}$ is calculated by a recursion from the previous $\boldsymbol{x}^{(k-1)}$. To summarize, the algorithm takes the following form

Algorithm LSQR

Initialize: $\boldsymbol{u}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2$, $\beta_1 = \|\boldsymbol{b}\|_2$ and set $v_0 = 0$,
for $k = 1, 2, \ldots$
  Define the kth approximate LSQ problem by generating
    $\boldsymbol{u}_{k+1}$, $\boldsymbol{v}_k$, α_k, β_{k+1} using the formulas:
      $\alpha_k \boldsymbol{v}_k = A^T \boldsymbol{u}_k - \beta_k \boldsymbol{v}_{k-1}$, $\beta_{k+1}\boldsymbol{u}_{k+1} = A\boldsymbol{v}_k - \alpha_k \boldsymbol{u}_k$,
    choosing α_k and β_{k+1} so that $\boldsymbol{v}_k$ and $\boldsymbol{u}_{k+1}$ are normalized.
  Solve the problem:
    Compute the QR factorization of B_k by updating that of the previous step.
    Compute $\boldsymbol{x}^{(k)}$ from $\boldsymbol{x}^{(k-1)}$ by a recursion.
  Estimate the residual and solution norms $\|\boldsymbol{r}_k\|_2$, $\|\boldsymbol{x}_k\|_2$, and $\left\|A^T \boldsymbol{r}_k\right\|_2$ and the condition of B_k.
  Check for convergence.
end

The cost per iteration is $3m + 5n$ flops, in addition to two matrix-vector multiplications (with A and A^T). Besides the storage for the matrix A, the algorithm needs 2 vectors of length m and 3 vectors of length n. One of the stopping criteria in [179] is specifically designed for singular or ill-conditioned problems: the iterations are stopped if $\text{cond}(B_k)$ is larger than a given tolerance.

A new formulation of the bidiagonalization process called LSMR was recently presented [80]. Compared to LSQR, this algorithm requires an additional n flops per iteration and one additional vector of length n. The advantage of LSMR is that both $\|\boldsymbol{r}_k\|_2$ and $\left\|A^T \boldsymbol{r}_k\right\|_2$ decay monotonically with k (while only $\|\boldsymbol{r}_k\|_2$ is monotonic for LSQR), thus making it safer to

terminate the algorithm as soon as the residual $A^T \boldsymbol{r}_k$ associated with the normal equations is sufficiently small.

For very ill-conditioned problems, instead of solving the original LSQ problem, there is the option in LSQR of solving a damped LSQ problem (equivalent to Tikhonov regularization, as we will see in Section 10):

$$\min_{\boldsymbol{x}} \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} \boldsymbol{x} - \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix} \right\|_2 .$$

The quantities $\boldsymbol{u}_{j+1}, \boldsymbol{v}_j, \alpha_j, \beta_{j+1}$ generated by the Golub-Kahan process are independent of λ, but $\boldsymbol{y}_k$ will now be the solution of a damped LSQ subproblem.

As LSQR is mathematically equivalent to the conjugate gradient algorithm applied to the normal equations for the LSQ problem, the convergence results in exact arithmetic are the same. This means that the LSQR iterates converge to the LSQ solution and the method should in theory take at most n steps to obtain the exact solution. However, it may take many fewer or many more, specially for ill-conditioned systems. As for CGLS the norms of the solution and the residual have monotonic behavior.

With respect to stability, we mention the following comment from Björck [21]: "The conclusion from both the theoretical analysis and the experimental evidence is that LSQR and CGLS are both well behaved and achieve a final accuracy consistent with a backward stable method."

Although LSQR has no relation to normal equations, as CGLS has, the differences in stability of both methods are very small.

6.4 Practicalities: preconditioning and stopping criteria

Preconditioning is a convergence acceleration technique that, in the LSQ case, is equivalent to a transformation of variables. For this purpose one chooses a nonsingular matrix S (the preconditioner), such that the modified problem $\min_y \left\| AS^{-1}\boldsymbol{y} - \boldsymbol{b} \right\|_2$, with $S\boldsymbol{x} = \boldsymbol{y}$, requires fewer iterations and less work. Two important requirements are that AS^{-1} be better conditioned than A and that it also be easy to recover $\boldsymbol{x}^*$ from the minimum $\boldsymbol{y}^*$ of the above-transformed problem.

Assuming for simplicity that the normal equations are nonsingular and therefore symmetric and positive definite, the best preconditioner would be $S = R_1$, the triangular Cholesky factor of $A^T A$, since then the condition number of the matrix AS^{-1} would be 1.

In general for LSQ problems, the following preconditioners are frequently chosen:

- $S = \text{diag}(\|\boldsymbol{a}_1\|_2, \ldots, \|\boldsymbol{a}_n\|_2)$, which amounts to a diagonal column scaling imposing unit length.

Incomplete factorizations such as

- $S = \widehat{R}$, where $A^T A = \widehat{R}^T \widehat{R} - E$; here the matrix $\widehat{R}$ is an incomplete Cholesky factor, leaving out inconvenient elements of the exact Cholesky factor but keeping $\|E\|_2$ small. For example, one can either define a convenient storage structure for R and ignore fill-in elements of the exact Cholesky factor or drop out elements that are small in magnitude. The various preconditioners define a spectrum of methods between the direct, where $S = R$ (the exact Cholesky factor when $E = 0$), through the incomplete Cholesky factor $\widehat{R}$, to the un-preconditioned iterative method when $S = I$.
- $S = \widehat{R}$, where $\widehat{R}$ comes from an approximate QR factor of A.
- $S = U\Pi_2^T$, where the factors are from an LU factorization $\Pi_1 A \Pi_2 = L \begin{pmatrix} U \\ 0 \end{pmatrix}$. L has unit diagonals and Π_1 is chosen to keep $|L_{ij}| \leq O(1)$. (Π_2 is less critical if A has column rank.)

It may be useful to remember the mathematical equivalence of LSQR and CGLS. The latter method minimizes the quadratic form

$$\min_x [(x^{(k)} - \boldsymbol{x})^T A^T A (\boldsymbol{x}^{(k)} - \boldsymbol{x})].$$

To precondition either of the two procedures can be considered intuitively as an attempt to stretch the quadratic form to make it more "spherical," so that the eigenvalues are closer to each other.

The simplest preconditioner, the diagonal matrix, is a scaling along the coordinate axis, whereas choosing $S^{-1} = (A^T A)^{-1}$ would be a scaling along the eigenvectors. Finding the best preconditioner depends on the problem and is still a research topic. A good survey can be found in [20].

Stopping criteria should always be chosen with great care so as to ensure the computation of a solution with sufficient accuracy (that, in the end, always depends on the application). For a good discussion of various stopping criteria we refer to [179]. Some general considerations are as follows:

1. Always specify a maximum number of iterations, in order to prevent an infinite loop.

2. Stop when the normal-equation residual's norm

$$\|A\,\boldsymbol{x}^{(k)} - \boldsymbol{b}\|_2$$

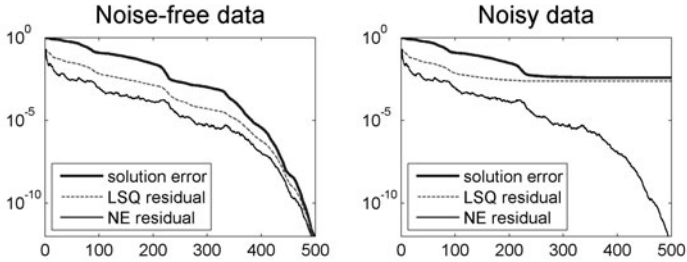

Figure 6.4.1: The relative norms of the errors, the LSQ residuals and the normal equation (NE) residuals, as a function of the number of iterations, for the WELL1850 geodetic surveying problem. Left: a system with a noise-free right-hand side $\boldsymbol{b}_{\text{exact}}$. Right: a system with a noisy right-hand side $\boldsymbol{b} = \boldsymbol{b}_{\text{exact}} + \boldsymbol{e}$.

is of the same size as the norm $\|\boldsymbol{e}\|_2$ of the vector of errors in the right-hand side, cf. Section 2.2.5. When this is the case, we have computed a solution $\boldsymbol{x}^{(k)}$ that fits the data as accurately as the errors allow (recall that we cannot achieve a zero LSQ residual for noisy data). It is not realistic to expect that the LSQ residual norm can get smaller than $\|\boldsymbol{e}\|_2$ and in fact, further iterations will lead to overfitting, i.e., modeling of the errors.

3. Stop when the norm of the normal-equation residual $\|A^T(A\,\boldsymbol{x}^{(k)} - \boldsymbol{b})\|_2$ is sufficiently small: In principle this norm should converge to zero, but rounding errors will usually prevent it. For well-conditioned problems, a small normal-equation residual is a sign that we have computed an accurate LSQ solution.

4. For the LSQR algorithm applied to ill-conditioned problems, stop when the condition number of the bidiagonal matrix B_k (or an estimate) exceeds a given threshold. This approach can be used to limit the sensitivity of the approximate LSQR solution $\boldsymbol{x}^{(k)}$ to data errors (at the cost of larger residual norms).

Example 73. ***Convergence of CGLS solutions.*** *This example illustrates the convergence of the iterates $\boldsymbol{x}^{(k)}$ of the CGLS algorithm (the LSQR iterates are practically identical for this problem). This LSQ problem arises in geodetic surveying and is available as problem "WELL1850" from Matrix Market [157], except that the noise-free right-hand side $\boldsymbol{b}_{\text{exact}}$ is modified so that the noise-free system is consistent, i.e., $A\,\boldsymbol{x}_{\text{exact}} = \boldsymbol{b}_{\text{exact}}$. The matrix A is 1850×712 and its condition number is 111.3 (so this is a rather small and well-conditioned system).*

Figure 6.4.1 shows the convergence for two systems. Left: for a system with the noise-free right-hand side $\boldsymbol{b}_{\text{exact}}$. *Right: with a noisy right-hand side* $\boldsymbol{b} = \boldsymbol{b}_{\text{exact}} + \boldsymbol{e}$, *where the perturbation* $\boldsymbol{e}$ *was scaled such that* $\|\boldsymbol{e}\|_2/\|\boldsymbol{b}_{\text{exact}}\|_2 = 3 \cdot 10^{-3}$. *We show:*

- *the relative error norm* $\|\boldsymbol{x}_{\text{exact}} - \boldsymbol{x}^{(k)}\|_2/\|\boldsymbol{x}_{\text{exact}}\|_2$,
- *the relative LSQ residual norm* $\|\boldsymbol{b} - A\,\boldsymbol{x}^{(k)}\|_2/\|\boldsymbol{b}_{\text{exact}}\|_2$ *and*
- *the relative normal-equation (NE) residual norm*

$$\|A^T(\boldsymbol{b} - A\,\boldsymbol{x}^{(k)})\|_2/\|\boldsymbol{x}_{\text{exact}}\|_2.$$

For the noise-free data we see that the solution error as well as both of the residuals converge to zero, clearly demonstrating that the iterates converge to the exact solution $\boldsymbol{x}_{\text{exact}}$, *which in this case is identical to the LSQ solution* $\boldsymbol{x}^*$. *For the noisy data, both the relative error norm and the relative LSQ residual norm level off, showing that the iterates now do not converge to* $\boldsymbol{x}_{\text{exact}}$ *as expected, due to the presence of the noise. In fact, the relative norm of the LSQ residual levels off at approximately* 3×10^{-3}, *as expected from the noise that we introduced. The relative NE residual norm converges to zero, showing that CGLS converges to a solution that satisfies the normal equations, i.e., the LSQ solution* $\boldsymbol{x}^*$ *for the problem.*

6.5 Block methods

There are a variety of classical and modern variations of block methods. We are interested here in methods that are suitable for parallel implementation. Very early on, Chazan and Miranker [43] devised a family of methods for solving large systems of linear equations under the name of chaotic relaxation with parallelization as the principal objective. The main characteristic of interest was that the methods were asynchronous, an important property in a network of heterogeneous computers or when the tasks are not all uniform in the amount of work. Synchronization points force a wait for all the processors to terminate and create the need for careful load balancing to avoid inefficiencies.

We describe such an algorithm for solving large linear LSQ problems. The idea is to partition the data into disjoint blocks and then select those variables that are best defined by each particular block. If the subproblems are small enough, then direct methods can be employed to solve them.

In Chapter 9 we extend these ideas to the nonlinear case and describe in detail the algorithm to determine the subgroup of variables associated with a given subset of equations. We also give there conditions for convergence of the (necessarily) iterative process that results. Two variants are discussed: a block Jacobi (BJ) and a block Gauss-Seidel (BGS) approach.

We allow repetition of variables in different blocks, and to prevent oscillations we use a running average for updating such repeated variables in the BGS method. For BJ a normal average is used after a full sweep of all the blocks. A user-chosen threshold is an integral part of the variable selection procedure. A smaller threshold allows more variables in the block and vice versa; thus the threshold itself acts as a regularization parameter for the least squares subproblems. Since we need to keep the blocks overdetermined, that can also be achieved by a judicious use of the threshold. Of course, if the global problem is very ill-conditioned the subproblems can use further regularization. For direct methods one can either use TSVD or Tikhonov regularization. Alternatively, one can use conjugate gradients for the blocks, with its own self-regularization by early termination.

Specifically, let the kth block of the least squares problem be written as

$$\min_{\boldsymbol{x}^{[k]}} \|\boldsymbol{r}^{[k]}\|_2^2 = \min_{\boldsymbol{x}^{[k]}} \sum_{i \in \mathsf{RI}_k} \left(\sum_{j \in \mathsf{CI}_k} a_{ij} x_j^{[k]} + \sum_{j \notin \mathsf{CI}_k} a_{ij} x_j - b_i \right)^2, \qquad (6.5.1)$$

where CI_k is the subset of indices of the variables that are active in block k, while the variables x_j with $j \notin \mathsf{CI}_k$ are kept fixed. The summation is over the set of data indices RI_k corresponding to observations in block k. The block iteration then consists in solving equation (6.5.1) for each block $k = 1, \ldots, K$. Since

$$\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2 = \min_{\boldsymbol{x}} \sum_{k=1}^{K} \|\boldsymbol{r}^{[k]}\|_2^2,$$

we expect that on termination of the block process we have an approximate solution of the global problem.

In a practical implementation on a cluster of processors, we assume that the current value of all the variables is held in a central location accessible to all the processors. If the values obtained in the solution of block k are used immediately to update the central repository, then that is a block Gauss-Seidel iteration, while if one waits until the end of the cycle to update all the variables, that will be a block Jacobi iteration, requiring a synchronization point at the end of each sweep over all the blocks.

In a sequential BGS iteration, the blocks would be visited in order, while in a parallel implementation we allow the block solves to arrive in an arbitrary order: that is the chaotic part. The theory provides for convergence, provided that each block is solved "often" enough, and it allows for occasional processor failures, an important feature.

We think that this algorithm is appropriate and will be most successful for problems in which both the data and the unknowns are "local" and essentially the problem can be decomposed into blocks that are only weakly coupled to variables not in the block (using a reasonable threshold).

In the literature there are a number of algorithms similar but not identical to this one, for which difficulties have been reported. Our experience with this method has been quite positive, so maybe some of the differences are important. Among the closest algorithm to ours is that of Steihaug and Yalcinkaya [229], where they partition the problem only in the variables, not allowing overlaps. Using sparse matrices from the Boeing collection, they show that convergence may deteriorate if old information pollutes the iteration. Bertsekas and his collaborators have done extensive work in this area [166, 218], although again their approach is somewhat different from the one described above.

A very interesting and extensive discussion of the solution of very large linear systems by combining block and iterative methods can be found in [52, 53]. It is a very good source for issues related to local clusters and geographically distributed ones (i.e., grid computing) and contains extensive experimentation and good pointers to available software. Many of those discussions are applicable to least squares problems.

This page intentionally left blank

Chapter 7

Additional Topics in Least Squares

In this chapter we collect some more specialized topics, such as problems with constraints, sensitivity analysis, total least squares and compressed sensing.

7.1 Constrained linear least squares problems

The inclusion of constraints in the linear least squares problem is often a convenient way to incorporate a priori knowledge about the problem. Linear equality constraints are the easiest to deal with, and their solution is an important part of solving problems with inequality constraints. Bound constraints and quadratic constraints for linear least squares are also considered in this chapter.

Least squares with linear constraints

Linear equality constraints (LSE)

The general form of a linear least squares problem with linear equality constraints is as follows: find a vector $\boldsymbol{x} \in \mathbb{R}^n$ that minimizes $\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2$ subject to the constraints $C^T\boldsymbol{x} = \boldsymbol{d}$, where C is a given $n \times p$ matrix with $p \leq n$ and $\boldsymbol{d}$ is a given vector of length p. This results in the LSE problem:

$$\boxed{\text{Problem LSE:} \quad \min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2 \quad \text{subject to} \quad C^T\boldsymbol{x} = \boldsymbol{d}.} \tag{7.1.1}$$

A solution exists if $C^T\boldsymbol{x} = \boldsymbol{d}$ is consistent, which is the case if $\operatorname{rank}(C) = p$. For simplicity, we assume that this is satisfied; Björck ([20], pp. 188) ex-

plains ways to proceed when there is no a priori knowledge about consistency. Furthermore, the minimizing solution will be unique if the augmented matrix

$$A_{\text{aug}} = \begin{pmatrix} C^T \\ A \end{pmatrix} \tag{7.1.2}$$

has full rank n.

The idea behind the different algorithms for the LSE problem is to reduce it to an unconstrained (if possible lower-dimensional) LSQ problem. To use Lagrange multipliers is of course an option, but we will instead describe two more direct methods using orthogonal transformations, each with different advantages.

One option is to reformulate the LSE problem as a weighted LSQ problem, assigning large weights (or penalties) for the constraints, thus enforcing that they are almost satisfied:

$$\min_{\boldsymbol{x}} \left\| \begin{pmatrix} \lambda C^T \\ A \end{pmatrix} \boldsymbol{x} - \begin{pmatrix} \lambda \boldsymbol{d} \\ \boldsymbol{b} \end{pmatrix} \right\|_2, \qquad \lambda \text{ large.} \tag{7.1.3}$$

Using the generalized singular value decomposition (GSVD), it can be proved that if $\boldsymbol{x}(\lambda)$ is the solution to (7.1.3) then $\|\boldsymbol{x}(\lambda) - \boldsymbol{x}\|_2 = \mathcal{O}(\lambda^{-2})$, so that in fact $\boldsymbol{x}(\lambda) \to \boldsymbol{x}$ as $\lambda \to \infty$. The LU factorization algorithm from Section 4.2 is well suited to solve this problem, because p steps of Gaussian elimination will usually produce a well-conditioned L matrix.

Although in principle only a general LSQ solver is needed, there are numerical difficulties because the matrix becomes poorly conditioned for increasing values of λ. However, a strategy described in [150], based on Householder transformations combined with appropriate row and column interchanges has been found to give satisfactory results. In practice, as [20] mentions, if one uses the LSE equations in the form (7.1.3), it is sufficient to apply a QR factorization with column permutations.

To get an idea of the size of the weight λ we refer to an example by van Loan described in [20], pp. 193. One can obtain almost 14 digits accuracy with a weight of $\lambda = 10^7$ using a standard QR factorization with permutations (e.g., MATLAB's function "qr"), if the constraints are placed in the first rows. Inverting the order in the equations, though, gives only 10 digits for the same weight λ and an increase in λ only degrades the computed solution. In addition, Björck [20] mentions a QR decomposition based on self-scaling Givens rotations that can be used without the "risk of overshooting the optimal weights."

Another commonly used algorithm directly eliminates some of the variables by using the constraints. The actual steps to solve problem (7.1.1) are:

1. Compute the QR factorization of C to obtain:

$$C = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \qquad \Longleftrightarrow \qquad C^T = \begin{pmatrix} R_1^T & 0 \end{pmatrix} Q^T.$$

2. Use the orthogonal matrix to define new variables:

$$\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = Q^T \boldsymbol{x} \qquad \Longleftrightarrow \qquad \boldsymbol{x} = Q \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} \tag{7.1.4}$$

and also to compute the value of the unknown p-vector $\boldsymbol{u}$ by solving the lower triangular system $R_1^T \boldsymbol{u} = \boldsymbol{d}$.

3. Introduce the new variables into the equation

$$\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2 = \left\|\boldsymbol{b} - AQQ^T\boldsymbol{x}\right\|_2 \equiv \left\|\boldsymbol{b} - \widetilde{A} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix}\right\|_2,$$

where the matrix $\widetilde{A} = AQ$ is partitioned according to the dimensions of $\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix}$: $\widetilde{A} = \begin{pmatrix} \widetilde{A}_1 & \widetilde{A}_2 \end{pmatrix}$, allowing the reduced $n-p$ dimensional LSQ problem to be written as

$$\min_{\boldsymbol{v}} \|(\boldsymbol{b} - \widetilde{A}_1 \boldsymbol{u}) - \widetilde{A}_2 \boldsymbol{v}\|_2. \tag{7.1.5}$$

4. Solve the unconstrained, lower-dimensional LSQ problem in (7.1.5) and obtain the original unknown vector $\boldsymbol{x}$ from (7.1.4).

This approach has the advantage that there are fewer unknowns in each system that need to be solved for and moreover the reformulated LSQ problem is better conditioned since, due to the interlacing property of the singular values: $\mathrm{cond}(\widetilde{A}_2) \leq \mathrm{cond}(\widetilde{A}) = \mathrm{cond}(A)$. The drawback is that sparsity will be destroyed by this process.

If the augmented matrix in (7.1.2) has full rank, one obtains a unique solution $\boldsymbol{x}_C^*$, if not, one has to apply a QR factorization with column permutations, while solving problem (7.1.5) to obtain the minimum-length solution vector.

The method of direct elimination compares favorably with another QR-based procedure, the null space method (see, for example, [20, 150]), both in numerical stability and in operational count.

Example 74. *Here we return to the linear prediction problem from Example 28, where we saw that* 4 *coefficients were sufficient to describe the particular signal used there. Hence, if we use* $n = 4$ *unknown LP coefficients, then we obtain a full-rank problem with a unique solution given by*

$$x^* = (\,-1,\, 0.096,\, -0.728,\, -0.096\,)^T.$$

If we want the prediction scheme in (2.3.3) to preserve the mean of the predicted signal, then we should add a linear constraint to the LSQ problem, forcing the prediction coefficients sum to zero, i.e., $\sum_{i=1}^{4} x_i = 0$. *In the above notation this corresponds to the linear equality constraint*

$$C^T \boldsymbol{x} = \boldsymbol{d}, \qquad C^T = (1,1,1,1), \qquad \boldsymbol{d} = 0.$$

Following the direct elimination process described above, this corresponds to the following steps for the actual problem.

1. *Compute the QR factorization of* C*:*

$$C = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = Q\,R = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

2. *Solve* $R_1^T \boldsymbol{u} = \boldsymbol{d} \Leftrightarrow 2u = 0 \Leftrightarrow u = 0$.

3. *Solve* $\min_{\boldsymbol{v}} \left\| \boldsymbol{b} - \widetilde{A}_2 \boldsymbol{v} \right\|_2$ *with*

$$\widetilde{A}_2 = A \begin{pmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & -0.5 \\ -0.5 & 0.5 & -0.5 \\ -0.5 & -0.5 & 0.5 \end{pmatrix},$$

   giving $\boldsymbol{v} = (\,-0.149,\, -0.755,\, 0.324\,)^T$.

4. *Compute the constrained solution*

$$\begin{aligned} \boldsymbol{x}_C^* &= Q \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} \\ &= \begin{pmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & -0.5 \\ -0.5 & 0.5 & -0.5 \\ -0.5 & -0.5 & 0.5 \end{pmatrix} \begin{pmatrix} -0.149 \\ -0.755 \\ 0.324 \end{pmatrix} = \begin{pmatrix} -0.290 \\ 0.141 \\ -0.465 \\ 0.614 \end{pmatrix}. \end{aligned}$$

It is easy to verify that the elements of $\boldsymbol{x}_C^*$ *sum to zero. Alternatively we can use the weighting approach in (7.1.3), with the row* λC^T *added on top of the matrix* A*; with* $\lambda = 10^2$, 10^4 *and* 10^8 *we obtain solutions almost identical to the above* $\boldsymbol{x}_C^*$*, with elements that sum to* $-1.72 \cdot 10^{-3}$, $-1.73 \cdot 10^{-7}$ *and* $-1.78 \cdot 10^{-15}$*, respectively.*

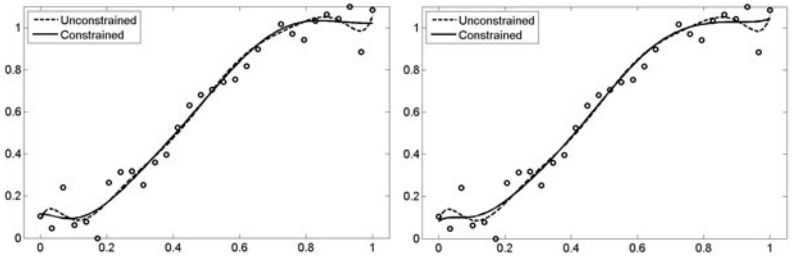

Figure 7.1.1: Constrained least squares polynomial fits ($m = 30$, $n = 10$). The unconstrained fit is shown by the dashed lines, while the constrained fit are shown by the solid lines. Left: equality constraints $M(\boldsymbol{x}, 0.5) = 0.65$ and $M'(\boldsymbol{x}, 0) = M'(\boldsymbol{x}, 1) = 0$. Right: inequality constraints $M'(\boldsymbol{x}, t_i) \geq 0$ for $i = 1, \ldots, m$.

Linear inequality constraints (LSI)

Instead of linear equality constraints we can impose linear inequality constraints on the least squares problem. Then the problem to be solved is

$$\boxed{\text{Problem LSI:} \quad \min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2 \quad \text{subject to} \quad \boldsymbol{l} \leq C^T \boldsymbol{x} \leq \boldsymbol{u},} \tag{7.1.6}$$

where the inequalities are understood to be component-wise.

There are several important cases extensively discussed in [20, 150], and Fortran subroutines are available from [88]. A good reference for this part is [91]. A constrained problem may have a minimizer of $\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2$ that is feasible, in which case it can be solved as an unconstrained problem. Otherwise a constrained minimizer will be located on the boundary of the feasible region. At such a solution, one or more constraints will be active, i.e., they will be satisfied with equality.

Thus the solver needs to find which constraints are active at the solution. If those were known a priori, then the problem could be solved as an equality constrained one, using one of the methods we discussed above. If not, then a more elaborate algorithm is necessary to verify the status of the variables and modify its behavior according to which constraints become active or inactive at any given time.

As we showed above, a way to avoid all this complication is to use penalty functions that convert the problem into a sequence of unconstrained problems. After a long hiatus, this approach has become popular again in the form of interior point methods [259]. It is, of course, not devoid of its own complications (principle of conservation of difficulty!).

Example 75. *This example illustrates how equality and inequality constraints can be used to control the properties of the fitting model* $M(\boldsymbol{x}, t)$,

using the rather noisy data ($m = 30$) shown in Figure 7.1.1 and a polynomial of degree 9 (i.e., $n = 10$). In both plots the dashed line shows the unconstrained fit.

Assume that we require that the model $M(\boldsymbol{x}, t)$ must interpolate the point $(t_{\text{int}}, y_{\text{int}}) = (0.5, 0.65)$, have zero derivative at the end points $t = 0$, and $t = 1$, i.e., $M'(\boldsymbol{x}, 0) = M'(\boldsymbol{x}, 1) = 0$. The interpolation requirement correspond to the equality constraint

$$(t_{\text{int}}^9, t_{\text{int}}^8, \ldots, t_{\text{int}}, 1)\, \boldsymbol{x} = y_{\text{int}},$$

while the constraint on $M'(\boldsymbol{x}, t)$ has the form

$$(9t^8, 8t^7, \ldots, 2t, 1, 0)\, \boldsymbol{x} = M'(\boldsymbol{x}, t).$$

Hence the matrix and the right-hand side for the linear constraints in the LSE problem (7.1.1) has the specific form

$$C^T = \begin{pmatrix} t_{\text{int}}^9 & t_{\text{int}}^8 & \cdots & t_{\text{int}}^2 & t_{\text{int}} & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ 9 & 8 & \cdots & 2 & 1 & 0 \end{pmatrix}, \qquad \boldsymbol{d} = \begin{pmatrix} y_{\text{int}} \\ 0 \\ 0 \end{pmatrix}.$$

The resulting constrained fit is shown as the solid line in the left plot in Figure 7.1.1.

Now assume instead that we require that the model $M(\boldsymbol{x}, t)$ be monotonically non-decreasing in the data interval, i.e., that $M'(\boldsymbol{x}, t) \geq 0$ for $t \in [0, 1]$. If we impose this requirement at the data points $t_1, \ldots, t_m$, we obtain a matrix C in the LSI problem (7.1.6) of the form

$$C^T = \begin{pmatrix} 9t_1^8 & 8t_1^7 & \ldots & 2t_1 & 0 \\ 9t_2^8 & 8t_2^7 & \ldots & 2t_2 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 9t_m^8 & 8t_m^7 & \ldots & 2t_m & 0 \end{pmatrix},$$

while the two vectors with the bounds are $\boldsymbol{l} = \boldsymbol{0}$ and $\boldsymbol{u} = \boldsymbol{\infty}$. The resulting monotonic fit is shown as the solid line in the right plot in Figure 7.1.1.

Bound constraints

It is worthwhile to discuss the special case of bounded variables, i.e., when $C = I$ in (7.1.6) – also known as box constraints. Starting from a feasible point (i.e., a vector $\boldsymbol{x}$ that satisfies the bounds), we iterate in the usual way for an unconstrained nonlinear problem (see Chapter 9), until a constraint is about to be violated. That identifies a face of the constraint box and a particular variable whose bound is becoming active. We set that variable to the bound value and project the search direction into the corresponding

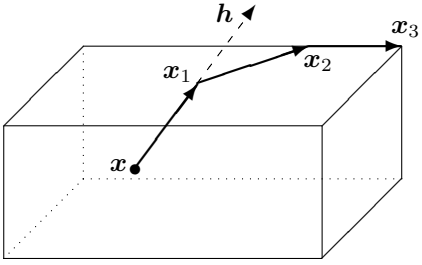

Figure 7.1.2: Gradient projection method for a bound-constrained problem with a solution at the upper right corner. From the starting point $\boldsymbol{x}$ we move along the search direction $\boldsymbol{h}$ until we hit the active constraint in the third coordinate, which brings us to the point $\boldsymbol{x}_1$. In the next step we hit the active constraint in the second coordinate, bringing us to the point $\boldsymbol{x}_2$. The third step brings us to the solution at $\boldsymbol{x}_3$, in which all three coordinates are at active constraints.

face, in order to continue our search for a constrained minimum on it. See Figure 7.1.2 for an illustration. If the method used is gradient oriented, then this technique is called the gradient projection method [210].

Example 76. *The use of bound constraints can sometimes have a profound impact on the LSQ solution, as illustrated in this example where we return to the CT problem from Example 72. Here we add noise* $\boldsymbol{e}$ *with relative noise levels* $\eta = \|\boldsymbol{e}\|_2/\|\boldsymbol{b}\|_2$ *equal to* 10^{-3}, $3 \cdot 10^{-3}$, 10^{-2} *and we enforce non-negativity constraints, i.e.,* $\boldsymbol{l} = \boldsymbol{0}$ *(and* $\boldsymbol{u} = \boldsymbol{\infty}$*). Figure 7.1.3 compares the LSQ solutions (bottom row) with the non-negativity constrained NNLS solutions (top row), and we see that even for the largest noise level* 10^{-2} *the NNLS solution includes recognizable small features – which are lost in the LS solution even for the smaller noise level* $3 \cdot 10^{-3}$.

Sensitivity analysis

Eldén [74] gave a complete sensitivity analysis for problem LSE (7.1.1), including perturbations of all quantities involved in this problem; here we specialize his results to the case where only the right-hand side $\boldsymbol{b}$ is perturbed. Specifically, if $\widetilde{\boldsymbol{x}}_C^*$ denotes the solution with the perturbed right-hand side $\boldsymbol{b} + \boldsymbol{e}$, then Eldén showed that

$$\|\boldsymbol{x}_C^* - \widetilde{\boldsymbol{x}}_C^*\|_2 \le \|A_C^\dagger\|_2 \, \|\boldsymbol{e}\|_2, \qquad A_C = A\,(I - (C^\dagger)^T C^T).$$

To derive a simpler expression for the matrix A_C consider the QR factorization $C = Q_1 R_1$ introduced above. Then $C^\dagger = R_1^{-1} Q_1^T$ and it follows

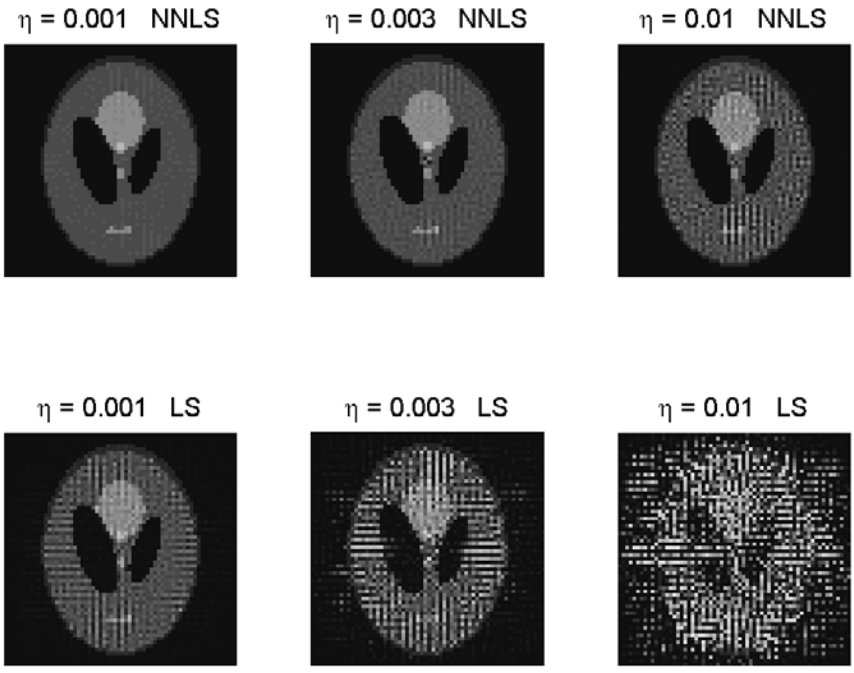

Figure 7.1.3: Reconstructions of the CT problem for three different relative noise levels $\eta = \|\boldsymbol{e}\|_2/\|\boldsymbol{b}\|_2$. Top: LSQ solutions with non-negativity constraints. Bottom: standard LSQ solutions.

that if $Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}$ then $I - (C^\dagger)^T C^T = I - Q_1 Q_1^T = Q_2 Q_2^T$ and hence $A_C = A\,Q_2 Q_2^T = \widetilde{A}_2 Q_2^T$. Moreover, we have $A_C^\dagger = Q_2 \widetilde{A}_2^\dagger$ and thus $\|A_C^\dagger\|_2 = \|\widetilde{A}_2^\dagger\|_2$. The following theorem then follows immediately.

Theorem 77. *Let $\boldsymbol{x}_C^*$ and $\widetilde{\boldsymbol{x}}_C^*$ denote the solutions to problem LSE (7.1.1) with right-hand sides $\boldsymbol{b}$ and $\boldsymbol{b} + \boldsymbol{e}$, respectively. Then, neglecting second-order terms,*

$$\frac{\|\boldsymbol{x}_C^* - \widetilde{\boldsymbol{x}}_C^*\|_2}{\|\boldsymbol{x}_C^*\|_2} \leq \operatorname{cond}(\widetilde{A}_2)\, \frac{\|\boldsymbol{e}\|_2}{\|\widetilde{A}_2\|_2 \|\boldsymbol{x}_C^*\|_2}.$$

This implies that the equality-constrained LS solution is typically less sensitive to perturbations, since the condition number of $\widetilde{A}_2 = AQ_2$ is less than or equal to the condition number of A.

Least squares with quadratic constraints (LSQI)

If we add a quadratic constraint to the least squares problem, then we obtain a problem of the form

$$\boxed{\text{Problem LSQI: } \min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2 \text{ subject to } \|\boldsymbol{d} - B\,\boldsymbol{x}\|_2 \leq \alpha,} \tag{7.1.7}$$

where $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times n}$. We assume that

$$\operatorname{rank}(B) = r \qquad \text{and} \qquad \operatorname{rank}\begin{pmatrix} A \\ B \end{pmatrix} = n,$$

which guarantees a unique solution of the LSQI problem. Least squares problems with quadratic constraints arise in many applications, including ridge regression in statistics, Tikhonov regularization of inverse problems, and generalized cross-validation; we refer to [121] for more details.

To facilitate the analysis it is convenient to transform the problem into "diagonal" form by using the generalized singular value decomposition (GSVD) from Section 3.3:

$$U^T A\,X = D_A,\ V B\,X = D_B,\ \boldsymbol{b} = U^T\boldsymbol{b},\ \widetilde{\boldsymbol{d}} = V^T\boldsymbol{d},\ \boldsymbol{x} = X\boldsymbol{y}.$$

The matrices D_A, D_B are diagonal with non-negative elements $\alpha_1, \alpha_2, ..., \alpha_n$ and $\beta_1, \beta_2, ..., \beta_q$, where $q = \min\{p, n\}$ and there are r values $\beta_i > 0$. The reformulated problem is now

$$\min_{\boldsymbol{y}} \left\|\widetilde{\boldsymbol{b}} - D_A\boldsymbol{y}\right\|_2^2 = \min_{\boldsymbol{y}} \left(\sum_{i=1}^{n} \left(\alpha_i y_i - \widetilde{b}_i\right)^2 + \sum_{n+1}^{m} \widetilde{b}_i^2 \right) \tag{7.1.8}$$

$$\text{subject to } \left\|\widetilde{\boldsymbol{d}} - D_B\boldsymbol{y}\right\|_2^2 = \sum_{i=1}^{r} \left(\beta_i y_i - \widetilde{d}_i\right)^2 + \sum_{r+1}^{p} \widetilde{d}_i^2 \leq \alpha^2. \tag{7.1.9}$$

A necessary and sufficient condition for the existence of a solution is that $\sum_{i=r+1}^{p} \widetilde{d}_i^2 \leq \alpha^2$. The way to solve problem (7.1.8)–(7.1.9) will depend on the size of the term $\sum_{i=r+1}^{p} \widetilde{d}_i^2$.

1. If $\sum_{r+1}^{p} \widetilde{d}_i^2 = \alpha^2$, the only $\boldsymbol{y}$ that can satisfy the constraints has as first r elements $y_i = \widetilde{d}_i/\beta_i$, $i = 1, \ldots, r$. The remaining elements y_i are defined from the minimization of (7.1.8). The minimum is attained if $\sum_{i=1}^{n} (\alpha_i y_i - \widetilde{b}_i)^2 = 0$ or is as small as possible, so that for $i = r+1, \ldots, n$ we set $y_i = \widetilde{b}_i/\alpha_i$ if $\alpha_i \neq 0$ and $y_i = 0$ otherwise.

2. If $\sum_{i=r+1}^{p} \widetilde{d}_i^2 < \alpha^2$, one could use the Lagrange multipliers method directly – cf. Appendix C.2.1 – but it is also possible to try a simpler approach to reach a feasible solution: define the vector $\boldsymbol{y}$ that minimizes (7.1.8), which implies choosing $y_i = \widetilde{b}_i/\alpha_i$ if $\alpha_i \neq 0$ as before. If $\alpha_i = 0$, then try to make the left-hand side of the constraints as small as possible by defining $y_i = \widetilde{d}_i/\beta_i$ if $\beta_i \neq 0$ or else $y_i = 0$.

3. If the resulting solution $\boldsymbol{y}$ is feasible, i.e., it satisfies the constraints (7.1.9), then $\boldsymbol{x} = X\,\boldsymbol{y}$ is the LSQI solution.

4. If not, look for a solution on the boundary of the feasible set, i.e., where the constraint is satisfied with equality. That is the standard form for the use of Lagrange multipliers, so the problem is now, for

$$\Phi(\boldsymbol{y}; \mu) = \left\| \widetilde{\boldsymbol{b}} - D_A \boldsymbol{y} \right\|_2^2 - \mu \left(\left\| \boldsymbol{d} - D_B \boldsymbol{y} \right\|_2^2 - \alpha^2 \right) \tag{7.1.10}$$

find $\boldsymbol{y}_\mu$ and μ so that $\nabla \Phi = 0$.

It can be shown that the solution $\boldsymbol{y}_\mu$ in step 4 satisfies the "normal equations"

$$(D_A^T D_A + \mu D_B^T D_B)\, \boldsymbol{y}_\mu = D_A^T \widetilde{\boldsymbol{b}} + \mu D_B^T \widetilde{\boldsymbol{d}}, \tag{7.1.11}$$

where μ satisfies the *secular equation*:

$$\chi(\mu) = \left\| \widetilde{\boldsymbol{d}} - D_B \boldsymbol{y}_\mu \right\|_2^2 - \alpha^2 = 0.$$

An iterative Newton-based procedure can be defined as follows. Starting from an initial guess μ_0, at each successive step calculate $\boldsymbol{y}_{\mu_i}$ from (7.1.11), then compute a Newton step for the secular equation obtaining a new value μ_{i+1}. It can be shown that this iteration is monotonically convergent to a unique positive root if one starts with an appropriate positive initial value and if instead of $\chi(\mu) = 0$ one uses the more convenient form

$$\frac{1}{\left\| \widetilde{\boldsymbol{d}} - D_B \boldsymbol{y}_\mu \right\|_2^2} - \frac{1}{\alpha^2} = 0.$$

Therefore the procedure can be used to obtain the unique solution of the LSQI problem. This is the most stable, but also the most expensive numerical algorithm. If instead of using the GSVD reformulation one works with the original equations (7.1.7), an analogous Newton-based method can be applied, in which the first stage at each step is

$$\min_{\boldsymbol{x}_\mu} \left\| \begin{pmatrix} A \\ \sqrt{\mu} B \end{pmatrix} \boldsymbol{x}_\mu - \begin{pmatrix} \boldsymbol{b} \\ \sqrt{\mu} \boldsymbol{d} \end{pmatrix} \right\|_2 .$$

Efficient methods for solution of this kind of least squares problem have been studied for several particular cases; see [20] for details.

7.2 Missing data problems

The problem of missing data occurs frequently in scientific research. It may be the case that in some experimental plan, where observations had to be made at regular intervals, occasional omissions arise. Examples would be clinical trials with incomplete case histories, editing of incomplete surveys or, as in the example given at the end of this section, gene expression microarray data, where some values are missing. Let us assume that the missing data are MCAR (missing completely at random), i.e., the probability that a particular data element is missing is unrelated to its value or any of the variables. For example, in the case of data arrays, independent of the column or row.

The appropriate technique for data imputation (a statistical term, meaning the estimation of missing values), will depend among other factors, on the size of the data set, the proportion of missing values and on the type of missing data pattern. If only a few values are missing, say, 1–5%, one could use a single regression substitution: i.e., predict the missing values using linear regression with the available data and assign the predicted value to the missing score. The disadvantage of this approach is that this information is only determined from the – now reduced – available data set. However, with MCAR data, any subsequent statistical analysis remains unbiased. This method can be improved by adding to the predicted value a residual drawn to reflect uncertainty (see [152], Chapter 4).

Other classic processes to fill in the data to obtain a complete set are as follows:

- Listwise deletion: omit the cases with missing values and work with the remaining set. It may lead to a substantial decrease in the available sample size, but the parameter estimates are unbiased.

- Hot deck imputation: replace the missing data with a random value drawn from the collection of data of similar participants. Although

widely used in some applications there is scarce information about the theoretical properties of the method.

- Mean substitution: substitute a mean of the available data for the missing values. The mean may be formed within classes of data. The mean of the resulting set is unchanged, but the variance is underestimated.

More computationally intensive approaches based on least squares and maximum-likelihood principles have been studied extensively in the past decades and a number of software packages that implement the procedures have been developed.

Maximum likelihood estimation

These methods rely on probabilistic modeling, where we wish to find the maximum likelihood (ML) estimate for the parameters of a model, including both the observed and the missing data.

Expectation-maximization (or EM) algorithm

One ML algorithm is the expectation-maximization algorithm. This algorithm estimates the model parameters iteratively, starting from some initial guess of the ML parameters, using, for example, a model for the listwise deleted data. Then follows a recursion until the parameters stabilize, each step containing two processes.

- E-step: the distribution of the missing data is estimated given the observed data and the current estimate of the parameters.

- M-step: the parameters are re-estimated to those with maximum likelihood, assuming the complete data set generated in the E-step.

Once the iteration has converged, the final maximum likelihood estimates of the regression coefficients are used to estimate the final missing data. It has been proved that the method converges, because at each step the likelihood is non-decreasing, until a local maximum is reached, but the convergence may be slow and some acceleration method must be applied. The global maximum can be obtained by starting the iteration several times with randomly chosen initial estimates.

For additional details see [149, 152, 220]. For software IBM SPSS: missing value analysis module. Also free software such as NORM is available from [163].

Multiple imputation (MI)

Instead of filling in a single value for each missing value, Rubin's multiple imputation procedure [152], replaces each missing value with a set of plausible values that represent the uncertainty about the right value to impute. Multiple imputation (MI) is a Monte Carlo simulation process in which a number of full imputed data sets are generated. Statistical analysis is performed on each set, and the results are combined [68] to produce an overall analysis that takes the uncertainty of the imputation into consideration. Depending on the fraction of missing values, a number between 3 and 10 sets must be generated to give good final estimates.

The critical step is the generation of the imputed data set. The choice of the method used for this generation depends on the type of the missing data pattern (see, for example, [132]). For monotone patterns, a parametric regression method can be used, where a regression model is fitted for the variable with missing values with other variables as covariates (there is a hierarchy of missingness: if z_b is observed, then z_a for $a < b$, is also observed). This procedure allows the incorporation of additional information into the model, for example, to use predictors that one knows are related to the missing data. Based on the resulting model, a new regression model is simulated and used to impute the missing values.

For arbitrary missing data patterns, a computationally expensive Markov Chain Monte Carlo (MCMC) method, based on an assumption of multivariate normality, can be used (see [220]). MI is robust to minor departures from the normality assumption, and it gives reasonable results even in the case of a large fraction of missing data or small size of the samples.

For additional information see [132, 149, 152]. For software see [132].

Least squares approximation

We consider here the data as a matrix A with m rows and n columns, which is approximated by a low-rank model matrix. The disadvantage is that no information about the data distribution is included, which may be important when the data belong to a complex distribution. Two types of methods are in common use: SVD based and local least squares imputations.

SVD-based imputation

In this method, the singular value decomposition is used to obtain a set of orthogonal vectors that are linearly combined to estimate the missing data values. As the SVD can only be performed on complete matrices, one works with an auxiliary matrix $A^{'}$ obtained by substituting any missing position in A by a row average. The SVD of $A^{'}$ is as usual $A^{'} = U\Sigma V^T$. We select first the k most significant right singular vectors of $A^{'}$.

Then, one estimates the missing ij value of A by first regressing the row i against the significant right eigenvectors and then using a linear combination of the regression coefficients to compute an estimate for the element ij. (Note that the j components of the right eigenvectors are not used in the regression.) This procedure is repeated until it converges, and an important point is that the convergence depends on the configuration of the missing entries.

Local least squares imputation

We will illustrate the method with the imputation of a DNA microarray. The so-called DNA microarray, or DNA chip, is a technology used to experiment with many genes at once. To this end, single strands of complementary DNA for the genes of interest – which can be many thousands – are immobilized on spots arranged in a grid ("array") on a support that will typically be a glass slide, a quartz wafer or a nylon membrane.

The data from microarray experiments are usually in the form of large matrices of expression levels of genes (gene expression is the process by which information from a gene is used in the synthesis of a functional gene product), under different experimental conditions and frequently with some values missing. Missing values occur for diverse reasons, including insufficient resolution, image corruption or simply due to dust or scratches on the slide. Robust missing value estimation methods are needed, since many algorithms for gene expression analysis require a complete matrix of gene array values.

One method, developed by Kim, Golub and Park [145] for estimating the missing values, is a least squares-based imputation approach using the concept of coherent genes. The method is a local least squares (LLS) algorithm, since only the genes with high similarity with the target gene, the one with incomplete values, are used. The coherent genes are identified using similarity measures based on the ℓ_2-norm or the Pearson correlation coefficient. Once identified, two approaches are used to estimate the missing values, depending on the relative sizes between the number of selected similar genes and the available experiments:

1. Missing values can be estimated either by representing the target gene with missing values as a linear combination of the similar genes.

2. The target experiment that has missing values can be represented as a linear combination of related experiments.

Denote with $G \in \mathbb{R}^{m\times n}$ a gene expression data matrix with m genes and n experiments and assume $m \ggg n$. The row $\boldsymbol{g}_i^T$ of G represents expressions of the ith gene in n experiments. Assume for now that only

one value is missing and it corresponds to the first position of the first gene, $G(1,1) = \boldsymbol{g}_1(1)$, denoted by α for simplicity.

Now, among the genes with a known first position value, either the k nearest-neighbors gene vectors are located using the ℓ_2-norm or the k most similar genes are identified using the Pearson correlation coefficient. Then, based on these k closest gene vectors, a matrix $A \in \mathbb{R}^{k\times(q-1)}$ and two vectors $\boldsymbol{b} \in \mathbb{R}^k$ and $\boldsymbol{w} \in \mathbb{R}^{(q-1)}$ are formed. The k rows of the matrix A consist of the k closest gene vectors with their first values deleted; q varies depending on the number of known values in these similar genes (i.e., the number of experiments recorded successfully). The elements of $\boldsymbol{b}$ consist of the first values of these gene vectors, and the elements of $\boldsymbol{w}$ are the $q-1$ known elements of $\boldsymbol{g}_1$.

When $k < q-1$, the missing value in the target gene is approximated using the same-position value of the nearest genes:

1. Solve the local least squares problem $\min_{\boldsymbol{x}} \left\|A^T\boldsymbol{x} - \boldsymbol{w}\right\|_2$.
2. Estimate the missing value as a linear combination of the first values of the coherent genes $\alpha = \boldsymbol{b}^T\boldsymbol{x}$.

When $k \geq q-1$, on the other hand, the missing value is estimated using the experiments:

1. Solve the local least squares problem $\min_{\boldsymbol{x}} \|A\,\boldsymbol{x} - \boldsymbol{b}\|_2$.
2. Estimate the missing value as a linear combination of values of experiments, not taking into account the first experiment in the gene $\boldsymbol{g}_1$, i.e., $\alpha = \boldsymbol{w}^T\boldsymbol{x}$.

An improvement is to add weights of similarity for the k nearest neighbors, leading to weighted LSQ problems. In the actual DNA microarray data, missing values may occur in several locations. For each missing value the arrays A, $\boldsymbol{b}$ and $\boldsymbol{w}$ are generated and the LLS solved. When building the matrix A for a missing value, the already estimated values of the gene are taken into consideration.

An interesting result, based on experiments with data from the Stanford Microarray Database (SMD), is that the most robust missing value estimation method is the one based on representing a target experiment that has a missing value as a linear combination of the other experiments. A program called LSimpute is available from the authors of [145].

There are no theoretical results comparing different imputation algorithms, but the test results of [145, 238] are consistent and suggest that the above-described method is more robust for noisy data and less sensitive to k, the number of nearest neighbors used, than the SVD method. The appropriate choice of the k closest genes is still a matter of trial and error, although some experiments with random matrices point to thresholds for it [156]. See also [250].

7.3 Total least squares (TLS)

The assumption used until now for the LSQ problem is that errors are restricted to the right-hand side $\boldsymbol{b}$, i.e., the linear model is $A\boldsymbol{x} = \boldsymbol{b} + \boldsymbol{r}$ where $\boldsymbol{r}$ is the residual. A new problem arises when the data matrix A is also not known exactly, so both A and $\boldsymbol{b}$ have random errors. For simplicity, the statistical hypothesis will be that the rows of the errors are independent and have a uniform distribution with zero mean and common variance (a more general case is treated in [106]). This leads to the total least squares (TLS) problem of calculating a vector $\boldsymbol{x}_{\mathrm{TLS}}$ so that the augmented residual matrix $(\, E \;\; \boldsymbol{r} \,)$ is minimized:

$$\boxed{\text{Problem TLS: } \min \left\| (\, E \quad \boldsymbol{r} \,) \right\|_{\mathrm{F}}^2 \text{ subject to } (A+E)\,\boldsymbol{x} = \boldsymbol{b} + \boldsymbol{r}.} \quad (7.3.1)$$

We note that

$$\|(\, E \quad \boldsymbol{r} \,)\|_F^2 = \|E\|_F^2 + \|\boldsymbol{r}\|_2^2.$$

The relation to ordinary least squares problems is as follows:

- In the least squares approximation, one replaces the inconsistent problem $A\,\boldsymbol{x} = \boldsymbol{b}$ by the consistent system $A\,\boldsymbol{x} = \boldsymbol{b}'$, where $\boldsymbol{b}'$ is the vector closest to $\boldsymbol{b}$ in range(A), obtained by minimizing the orthogonal distance to range(A).
- In the total least squares approximation, one goes further and replaces the inconsistent problem $A\,\boldsymbol{x} = \boldsymbol{b}$ by the consistent system $A'\,\boldsymbol{x} = \boldsymbol{b}'$, finding the closest A' and $\boldsymbol{b}'$ to A and $\boldsymbol{b}$, respectively, by minimizing simultaneously the sum of squared orthogonal distances from the columns $\boldsymbol{a}_i$ to $\boldsymbol{a}_i'$ and $\boldsymbol{b}$ to $\boldsymbol{b}'$.

Example 78. *To illustrate the idea behind TLS we consider the following small problem:*

$$A = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The TLS solution $\boldsymbol{x}_{\mathrm{TLS}} = 1.618$ *is obtained with*

$$E = \begin{pmatrix} -0.276 \\ 0.447 \end{pmatrix}, \qquad \boldsymbol{r} = \begin{pmatrix} 0.171 \\ -0.276 \end{pmatrix}.$$

Figure 7.3.1 illustrates the geometrical aspects of this simple problem. We see that the perturbed right-hand side $\boldsymbol{b}+\boldsymbol{r}$ *and the perturbed first (and only) column* $A + E$ *are both orthogonal projections of* $\boldsymbol{b}$ *and* A*, respectively, on the subspace* range$(A + E)$*. The perturbations* $\boldsymbol{r}$ *and* E *are orthogonal to* $\boldsymbol{b}$ *and* A*, respectively, and they are chosen such that their lengths are minimal. Then* $\boldsymbol{x}_{\mathrm{TLS}}$ *is the ratio between the lengths of these two projections.*

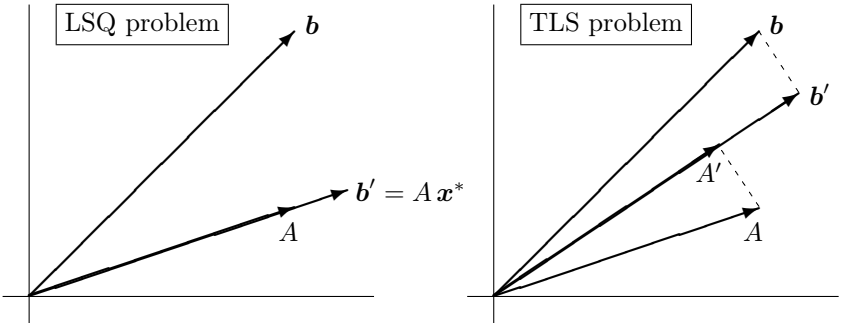

Figure 7.3.1: Illustration of the geometry underlying the LSQ problem (left) and the TLS problem (right). The LSQ solution $\boldsymbol{x}^*$ is chosen such that the residual $\boldsymbol{b}' - \boldsymbol{b}$ (the dashed line) is orthogonal to the vector $\boldsymbol{b}' = A\,\boldsymbol{x}^*$. The TLS solution $\boldsymbol{x}_{\mathrm{TLS}}$ is chosen such that the the residuals $\boldsymbol{b}' - \boldsymbol{b}$ and $A' - A$ (the dashed lines) are orthogonal to $\boldsymbol{b}' = A'\,\boldsymbol{x}_{\mathrm{TLS}}$ and A', respectively.

We will only consider the case when $\mathrm{rank}(A) = n$ and $\boldsymbol{b} \notin \mathrm{range}(A)$. The trivial case when $\boldsymbol{b} \in \mathrm{range}(A)$ means that the system is consistent, $(E\ \boldsymbol{r}) = 0$ and the TLS solution is identical to the LSQ solution.

We note that the rank-deficient matrix case, $\mathrm{rank}(A) < n$, has in principle only a solution in the trivial case $\boldsymbol{b} \in \mathrm{range}(A)$. In the general case it is treated by reducing the TLS problem to a smaller, full-rank problem using column subset selection techniques. More details are given in ([239], section 3.4). The work of Paige and Strakoš [180] on core problems is also relevant here. The following example taken from [105], p. 595, illustrates this case.

Example 79. *Consider the problem defined by*

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

The rank of A is 1 and the matrix is rank deficient, with $\boldsymbol{b} \notin \mathrm{range}(A)$, so there is no solution of the problem as is. Note that

$$E_\varepsilon = \begin{pmatrix} 0 & 0 \\ 0 & \varepsilon \\ 0 & \varepsilon \end{pmatrix}$$

is a perturbation, such that for any $\varepsilon \neq 0$ we have $\boldsymbol{b} \in \mathrm{range}(A + E_\varepsilon)$, but there is no smallest $\|E_\varepsilon\|_{\mathrm{F}}$.

The TLS problem and the singular value decomposition

Let us rewrite the TLS problem as a system:

$$\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} + \begin{pmatrix} E & \boldsymbol{r} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} = \boldsymbol{0},$$

or equivalently

$$C\,\boldsymbol{z} + F\,\boldsymbol{z} = \boldsymbol{0} \qquad \text{with} \qquad \boldsymbol{z} = \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix},$$

where $C = \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix}$ and $F = \begin{pmatrix} E & \boldsymbol{r} \end{pmatrix}$ are matrices of size $m \times (n+1)$. We seek a solution $\boldsymbol{z}$ to the homogeneous equation

$$(C + F)\,\boldsymbol{z} = \boldsymbol{0} \tag{7.3.2}$$

that minimizes $\|F\|_{\mathrm{F}}^2$. For the TLS problem to have a non-trivial solution $\boldsymbol{z}$, the matrix $C + F$ must be singular, i.e., $\operatorname{rank}(C + F) < n + 1$. To attain this, the SVD

$$C = \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} = U \Sigma V^T = \sum_{i=1}^{n+1} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

can be used to determine an acceptable perturbation matrix F. The singular values of A, here denoted by $\sigma_1' \geq \sigma_2' \geq \ldots \geq \sigma_n'$, will also enter in the discussion.

The solution technique is easily understood in the case when the singular values of C satisfy $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n > \sigma_{n+1}$, i.e., when the smallest singular value is isolated. Since we are considering the full-rank, non-trivial case, we also have $\sigma_{n+1} > 0$.

From the Eckart-Young-Mirski Theorem 43, the matrix nearest to C with a rank lower than $n + 1$ is at distance σ_{n+1} from C, and it is given by $\sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$. Thus, selecting $C + F = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ implies choosing a perturbation $F = -\sigma_{n+1} \boldsymbol{u}_{n+1} \boldsymbol{v}_{n+1}^T$ with minimal norm: $\|F\|_{\mathrm{F}} = \sigma_{n+1}$.

The solution $\boldsymbol{z}$ is now constructed using the fact that the $\boldsymbol{v}_i$ are orthonormal, and therefore $(C + F)\,\boldsymbol{v}_{n+1} = \boldsymbol{0}$. Thus, a general solution of the TLS problem is obtained by scaling the right singular vector $\boldsymbol{v}_{n+1}$ corresponding to the smallest singular value, in order to enforce the condition that $z_{n+1} = -1$:

$$z_i = \frac{v_{i,n+1}}{-v_{n+1,n+1}}, \qquad i = 1, 2, \ldots, n+1. \tag{7.3.3}$$

This is possible *provided that* $v_{n+1,n+1} \neq 0$. If $v_{n+1,n+1} = 0$, a solution does not exist and the problem is called *nongeneric*.

A theorem proved in [106] gives sufficient conditions for the existence of a unique solution. If the smallest singular value σ'_n of the full-rank matrix A is strictly larger than the smallest singular value of $(\; A \quad \boldsymbol{b} \;) = C$, σ_{n+1}, then $v_{n+1,n+1} \neq 0$ and a unique solution exists.

Theorem 80. *Denote by $\sigma_1, \ldots, \sigma_{n+1}$ the singular values of the augmented matrix $(\; A \quad \boldsymbol{b} \;)$ and by $\sigma'_1, \ldots, \sigma'_n$ the singular values of the matrix A. If $\sigma'_n > \sigma_{n+1}$, then there is a TLS correction matrix $(\; E \quad \boldsymbol{r} \;) = -\sigma_{n+1} u_{n+1} v_{n+1}^T$ and a unique solution vector*

$$\boxed{\boldsymbol{x}_{\mathrm{TLS}} = -(v_{1,n+1}, \ldots, v_{n,n+1})^T / v_{n+1,n+1}} \tag{7.3.4}$$

that solves the TLS problem.

Moreover, there are closed-form expressions for the solution:

$$\boldsymbol{x}_{\mathrm{TLS}} = (A^T A - \sigma_{n+1}^2 I)^{-1} A^T \boldsymbol{b}$$

and the residual norm:

$$\|A\,\boldsymbol{x}_{\mathrm{TLS}} - \boldsymbol{b}\|_2^2 = \sigma_{n+1}^2 \left(1 + \sum_{i=1}^{n} \frac{\left(\boldsymbol{u}_i^T \boldsymbol{b}\right)^2}{(\sigma'_i)^2 - \sigma_{n+1}^2} \right).$$

The following example, taken from pp. 179 in [20], illustrates the difficulty associated with the nongeneric case in terms of the SVDs.

Example 81. *Consider the augmented matrix:*

$$(\; A \quad \boldsymbol{b} \;) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \qquad \text{where} \qquad A = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The smallest singular value of A is $\sigma'_1 = 1$, and for the augmented matrix is $\sigma_2 = 1$. So here $\sigma'_n = \sigma_{n+1}$, and $v_{n+1,n+1} = 0$, and therefore no solution exists. The formal algorithm would choose the perturbation matrix from the dyadic form for $A + E = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$; then

$$(\; A + E \quad \boldsymbol{b} + \boldsymbol{r} \;) = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}$$

and it is easy to see that $\boldsymbol{b} + \boldsymbol{r} \notin \mathrm{range}(A + E)$, and there is no solution.

There are further complications with the nongeneric case, but we shall not go into these issues here primarily because they seem to have been resolved by the recent work of Paige and Strakoš [180].

In [239] p. 86, conditions for the existence and uniqueness of solutions are given, as well as closed form expressions for these solutions. In [239] p. 87, a general algorithm for the TLS problem is described. It solves

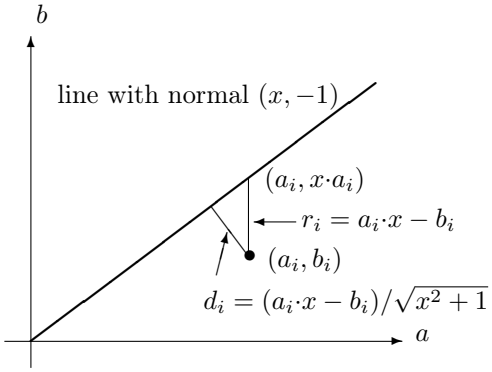

Figure 7.3.2: Illustration of LSQ and TLS for the case $n = 1$. The ith data point is (a_i, b_i) and the point vertically above it on the line is $(a_i, a_i{\cdot}x)$, hence the vertical distance is $r_i = a_i{\cdot}x - b_i$. The orthogonal distance to the line is $d_i = (x{\cdot}a_i - b_i)/\sqrt{x^2+1}$.

any generic and nongeneric TLS problem, including the case with multiple right-hand sides. The software developed by van Huffel is available from Netlib [263]. Subroutine PTLS solves the TLS problem by using the partial singular value decomposition (PSVD) mentioned in Section 5.6, thereby improving considerably the computational efficiency with respect to the classical TLS algorithm. A large-scale algorithm based on Rayleigh quotient iteration is described in [25]. See [155] for a recent survey of TLS methods.

Geometric interpretation

It is possible to give a geometric interpretation of the difference between the fits using the least squares and total least squares methods. Define the rows of the matrix $\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix}$ as the m points $\boldsymbol{c}_i = (a_{i1}, \ldots, a_{in}, b_i)^T \in \mathbb{R}^{n+1}$, to which we try to fit the linear subspace $\mathcal{S}_{\boldsymbol{x}}$ of dimension n that is orthogonal to the vector $\boldsymbol{z} = \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix}$:

$$\mathcal{S}_{\boldsymbol{x}} = \operatorname{span}\left\{ \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} \right\}^{\perp} = \left\{ \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{pmatrix} \middle| \; \boldsymbol{a} \in \mathbb{R}^n, \; \boldsymbol{b} \in \mathbb{R}, \; \boldsymbol{b} = \boldsymbol{x}^T \boldsymbol{a} \right\}.$$

In LSQ, one minimizes

$$\left\| \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} \right\|_2^2 = \sum_{k=1}^{m} \left(\boldsymbol{c}_k^T \boldsymbol{z} \right)^2 \qquad \boldsymbol{z} = \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix},$$

which measures the sum of squared distances along the $n+1$-coordinate axis from the points $\boldsymbol{c}_k$ to the subspace $\mathcal{S}_{\boldsymbol{x}}$ (already known as the residuals r_i). For the case $n = 1$, where $A = (a_1, a_2, \ldots a_m)^T$ and $\boldsymbol{x} = x$ (a single unknown), the LSQ approximation minimizes the *vertical* distance of the points (a_i, b_i) to the line through the origin with slope x (whose normal vector is $(x, -1)^T$); see Figure 7.3.2.

The TLS approach can be formulated as an equivalent problem.

From the SVD of $\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix}$ and the definition of matrix 2-norm we have

$$\left\| \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} \boldsymbol{z} \right\|_2 / \left\| \boldsymbol{z} \right\|_2 \geq \sigma_{n+1},$$

for any nonzero $\boldsymbol{z}$. If σ_{n+1} is isolated and the vector $\boldsymbol{z}$ has unit norm, then $\boldsymbol{z} = \pm \boldsymbol{v}_{n+1}$ and the inequality becomes an equality $\left\| \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} \boldsymbol{z} \right\|_2^2 = \sigma_{n+1}^2$. So, in fact the TLS problem consists of finding a vector $\boldsymbol{x} \in \mathbb{R}^n$ such that

$$\frac{\left\| \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} \right\|_2^2}{\left\| \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} \right\|_2^2} = \frac{\left\| A\,\boldsymbol{x} - \boldsymbol{b} \right\|_2^2}{\left\| \boldsymbol{x} \right\|_2^2 + 1} = \sigma_{n+1}^2. \tag{7.3.5}$$

The left-hand quantity is $\sum_{i=1}^{m} \frac{(\boldsymbol{a}_i^T \boldsymbol{x} - b_i)^2}{\boldsymbol{x}^T \boldsymbol{x} + 1}$, where the ith term is the square of the *true* Euclidean (or orthogonal) distance from $\boldsymbol{c}_i$ to the nearest point in the subspace $\mathcal{S}_{\boldsymbol{x}}$. Again, see Figure 7.3.2 for an illustration in the case $n = 1$.

The equivalent TLS formulation $\min_{\boldsymbol{x}} \left\| A\,\boldsymbol{x} - \boldsymbol{b} \right\|_2^2 / (\left\| \boldsymbol{x} \right\|_2^2 + 1)$ is convenient for regularizing the TLS problem additively; see next section. For more details see Section 10.2.

Further aspects of TLS problems

The sensitivity of the TLS problem

A study of the sensitivity of the TLS problem when there is a unique solution, i.e., when $\sigma'_n > \sigma_{n+1}$, is given in [106]. The starting point for the analysis is the formulation of the TLS problem as an eigenvalue problem for $C^T C = \sum_{i=1}^{n+1} \boldsymbol{v}_i \sigma_i^2 \boldsymbol{v}_i^T$ (i.e., σ_i^2 are the eigenvalues of $C^T C$ and $\boldsymbol{v}_i$ are the corresponding eigenvectors). The main tool used is the singular value interlacing property, Theorem 4 from Appendix B. Thus, if $\boldsymbol{x} \in \mathbb{R}^n$ and

$$C^T C \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} = \sigma_{n+1}^2 \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix},$$

then $\boldsymbol{x}$ solves the TLS problem.

One interesting result is that the total least squares problem can be considered as a *de-regularization* process, which is apparent when looking at the first row of the above eigenvalue problem:

$$(A^T A - \sigma_{n+1}^2 I)\, \boldsymbol{x}_{\mathrm{TLS}} = A^T \boldsymbol{b}.$$

This implies that the TLS problem is *worse conditioned* than the associated LSQ problem.

An upper bound for the difference of the LSQ and TLS solutions is

$$\frac{\|\boldsymbol{x}_{\mathrm{TLS}} - \boldsymbol{x}^*\|_2}{\|\boldsymbol{x}*\|_2} \leq \frac{\sigma_{n+1}^2}{\sigma_n'^{\,2} - \sigma_{n+1}^2},$$

so that $\sigma_n'^{\,2} - \sigma_{n+1}^2$ measures the sensitivity of the TLS problem. This suggests that the TLS solution is unstable (the bounds are large) if $\sigma_n^{'}$ is close to σ_{n+1}, for example, if σ_{n+1} is (almost) a multiple singular value.

Stewart [231] proves that up to second-order terms in the error, $\boldsymbol{x}_{\mathrm{TLS}}$ is insensitive to column scalings of A. Thus, unlike ordinary LSQ problems, TLS cannot be better conditioned by column scalings.

The mixed LS-TLS problem

Suppose that only some of the columns of A have errors. This leads to the model

$$\min_{E,\boldsymbol{r}} (\|E_2\|_{\mathrm{F}}^2 + \|\boldsymbol{r}\|_2^2) \quad \text{subject to} \quad (\ A_1 \quad A_2 + E_2\ )\, \boldsymbol{x} = \boldsymbol{b} + \boldsymbol{r}, \qquad (7.3.6)$$

with $A_1 \in \mathbb{R}^{m \times n_1}$ and $A_2, E_2 \in \mathbb{R}^{m \times n_2}$. This *mixed LS-TLS problem* formulation encompasses the LSQ problem (when $n_2 = 0$), the TLS problem (when $n_1 = 0$), as well as a combination of both.

How can one find the solution to (7.3.6)? The basic idea, developed in [95], is to use a QR factorization of $(\ A \quad \boldsymbol{b}\ )$ to transform the problem to a block structure, thereby reducing it to the solution of a smaller TLS problem that can be solved independently, plus an LSQ problem. First, compute the QR factorization $(\ A \quad \boldsymbol{b}\ ) = Q\, R$ with

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \qquad R_{11} \in \mathbb{R}^{n_1 \times n_1}, \quad R_{22} \in \mathbb{R}^{(n_2+1)\times(n_2+1)}.$$

Note that $\|F\|_{\mathrm{F}} = \|Q^T F\|_{\mathrm{F}}$ and therefore the constraints on the original and transformed systems are equivalent. Now compute the SVD of R_{22} and let $\boldsymbol{z}_2 = \boldsymbol{v}_{n_2+1}$ be the rightmost singular vector. We can then set $F_{12} = 0$ and solve the standard least squares problem

$$R_{11}\, \boldsymbol{z}_1 = -R_{12}\, \boldsymbol{z}_2.$$

Note that, as the TLS part of $\boldsymbol{z}$ has been obtained already and therefore F_{22} is minimized, there is no loss of generality when $F_{12} = 0$. Finally, compute the mixed LS-TLS solution as

$$\boldsymbol{x} = -\boldsymbol{z}(1:n)/\boldsymbol{z}(n+1), \qquad \boldsymbol{z} = \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix}.$$

7.4 Convex optimization

In an earlier section of this chapter we have already met a convex optimization problem: least squares with quadratic constraints. But there are many more problems of interest and some of them are related to the subject of this book.

Convex optimization has gained much attention in recent years, thanks to recent advances in algorithms to solve such problems. Also, the efforts of Stephen Boyd and his group at Stanford have helped to popularize the subject and provide invaluable software to solve all sizes of problems. One can say now that convex optimization problems have reached the point where they can be solved as assuredly as the basic linear algebra problems of the past.

One of the reasons for the success is that, although nonlinear, strictly convex problems have unique solutions. General nonlinear programming problems can have goal landscapes that are very bumpy and therefore it can be hard to find a desired optimal point and even harder to find a global one.

An important contribution in recent times has been in producing tools to identify convexity and to find formulations of problems that are convex. These problems look like general nonlinear programming problems, but the objective function $f(\boldsymbol{x})$ and the constraints $g_i(\boldsymbol{x})$ and $h_i(\boldsymbol{x})$ are required to be convex:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leq 0, \quad i = 1, ..., m \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, ..., p. \end{aligned}$$

In the past 30 years several researchers discovered that by adding a property (self-concordance) to these problems made it possible to use interior point methods via Newton iterations that were guaranteed to converge in polynomial time [167]. That was the breakthrough that allows us now to solve large problems in a reasonable amount of computer time and has provided much impetus to the application of these techniques to an ever increasing set of engineering and scientific problems.

A notable effort stemming from the research group of S. Boyd at Stanford is M. Grant's cvx MATLAB package, available from http://cvxr.com. van den Berg and Friedlander [240, 241] have also some interesting contributions, including efficient software for the solution of medium- and large-scale convex optimization problems; we also mention the Python software cvxopt by Dahl and Vandenberghe [56]. A comprehensive book on the subject is [30].

7.5 Compressed sensing

The theory and praxis of compressed sensing has become an important subject in the past few years. Compressed sensing is predicated on the fact that we often collect too much data and then compress it, i.e., a good deal of that data is unnecessary in some sense. Think of a mega-pixel digital photography and its JPEG compressed version. Although this is a lossy compression scheme, going back and forth allows for almost perfect reconstruction of the original image. Compressed sensing addresses the question: is it possible to collect much less data and use reconstruction algorithms that provide a perfect image? The answer in many cases is yes.

The idea of compressed sensing is that solution vectors can be represented by much fewer coefficients if an appropriate basis is used, i.e., they are sparse in such a base, say, of wavelets [164]. Following [34, 35] we consider the general problem of reconstructing a vector $\boldsymbol{x} \in \mathbb{R}^N$ from linear measurements, $y_k = \boldsymbol{\varphi}_k^T \boldsymbol{x}$ for $k = 1, \ldots, K$. Assembling all these measurements in matrix form we obtain the relation: $\boldsymbol{y} = \Phi\, \boldsymbol{x}$, where $\boldsymbol{y} \in \mathbb{R}^K$ and $\boldsymbol{\varphi}_k$ are the rows of the matrix Φ.

The important next concept is that compressed sensing attempts to reconstruct the original vector $\boldsymbol{x}$ from a number of samples smaller than what the Nyquist-Shannon theorem says is possible. The latter theorem states that if a signal contains a maximum frequency f (i.e., a minimum wavelength $w = 1/f$), then in order to reconstruct it exactly it is necessary to have samples in the time domain that are spaced no more than $1/(2f)$ units apart. The key to overcoming this restriction in the compressed sensing approach is to use nonlinear reconstruction methods that combine l_2 and l_1 terms.

We emphasize that in this approach the system $\boldsymbol{y} = \Phi\, \boldsymbol{x}$ is underdetermined, i.e., the matrix Φ has more columns than rows. It turns out that if one assumes that the vector $\boldsymbol{x}$ is sparse in some basis B, then solving the following convex optimization problem:

$$\min_{\widetilde{\boldsymbol{x}}} \|\widetilde{\boldsymbol{x}}\|_1 \quad \text{subject to} \quad \Phi B\, \widetilde{\boldsymbol{x}} = \boldsymbol{y}, \tag{7.5.1}$$

reconstructs $\boldsymbol{x} = B\widetilde{\boldsymbol{x}}$ exactly with high probability. The precise statement

is as follows.

Theorem 82. *Assume that $\boldsymbol{x}$ has at most S nonzero elements and that we take K random linear measurements, where K satisfies*

$$K \geqslant C \cdot S \cdot \log N,$$

where $C = 22(\delta+1)$ and $\delta > 0$. Then, solving (7.5.1) reconstructs $\boldsymbol{x}$ exactly with probability greater than $1 - O(N^{-\delta})$.

The "secret" is that the Nyquist-Shannon theorem talks about a whole band of frequencies, from zero to the highest frequency, while here we are considering a sparse set of frequencies that may be in small disconnected intervals. Observe that for applications of interest, the dimension N in problem (7.5.1) can be very large and there resides its complexity. This problem is convex and can be reduced to a linear programming one. For large N it is proposed to use an interior-point method that basically solves the optimality conditions by Newton's method, taking care of staying feasible throughout the iteration. E. Candes (Caltech, Stanford) offers a number of free software packages for this type of applications. Some very good recent references are [240, 241].

As usual, in the presence of noise and even if the constraints in (7.5.1) are theoretically satisfied, it is more appropriate to replace that problem by $\min_{\widetilde{\boldsymbol{x}}} \|\widetilde{\boldsymbol{x}}\|_1$ subject to $\|\Phi\,\widetilde{\boldsymbol{x}} - \boldsymbol{y}\|_2 \leq \alpha$, where the positive parameter α is an estimate of the data noise level.

It is interesting to notice the connection with the basic solution of rank-deficient problems introduced in Section 2.3. Already in 1964 J. B. Rosen wrote a paper [209] about its calculation and in [193] an algorithm and implementation for calculating the minimal and basic solutions was described. That report contains extensive numerical experimentation and a complete Algol program. Unfortunately, the largest matrix size treated there was 40 and therefore without further investigation that algorithm may not be competitive today for these large problems. Basic solutions are used in some algorithms as initial guesses to solve this problem, where one generally expects many fewer nonzero elements in the solution than the rank of the matrix.

This page intentionally left blank

Chapter 8

Nonlinear Least Squares Problems

So far we have discussed data fitting problems in which all the unknown parameters appear linearly in the fitting model $M(\boldsymbol{x}, t)$, leading to linear least squares problems for which we can, in principle, write down a closed-form solution. We now turn to nonlinear least squares problems (NLLSQ) for which this is not true, due to (some of) the unknown parameters appearing nonlinearly in the model.

8.1 Introduction

To make precise what we mean by the term "nonlinear" we give the following definitions. A parameter α of the function f appears nonlinearly if the derivative $\partial f/\partial\alpha$ is a function of α. A parameterized fitting model $M(\boldsymbol{x}, t)$ is nonlinear if at least one of the parameters in $\boldsymbol{x}$ appear nonlinearly. For example, in the exponential decay model

$$M(x_1, x_2, t) = x_1 e^{-x_2 t}$$

we have $\partial M/\partial x_1 = e^{-x_2 t}$ (which is independent of x_1, so is linear in it) and $\partial M/\partial x_2 = -t\, x_1 e^{-x_2 t}$ (which depends on x_2) and thus M is a nonlinear model with the parameter x_2 appearing nonlinearly. We start with a few examples of nonlinear models.

Example 83. ***Gaussian model.*** *All measuring devices somehow influence the signal that they measure. If we measure a time signal g_{mes}, then we can often model it as a convolution of the true signal $g_{\text{true}}(t)$ and an*

instrument response function $\Gamma(t)$*:*

$$g_{\text{mes}}(t) = \int_{-\infty}^{\infty} \Gamma(t-\tau)\, g_{\text{true}}(\tau)\, d\tau.$$

A very common model for $\Gamma(t)$ *is the non-normalized Gaussian function*

$$M(\boldsymbol{x}, t) = x_1 e^{-(t-x_2)^2/(2x_3^2)}, \tag{8.1.1}$$

where x_1 *is the amplitude,* x_2 *is the time shift and* x_3 *determines the width of the Gaussian function. The parameters* x_2 *and* x_3 *appear nonlinearly in this model. The Gaussian model also arises in many other data fitting problems.*

Example 84. ***Rational model.*** *Another model function that often arises in nonlinear data fitting problems is the rational function, i.e., a quotient of two polynomials of degree* p *and* q*, respectively,*

$$M(\boldsymbol{x}, t) = \frac{P(t)}{Q(t)} = \frac{x_1 t^p + x_2 t^{p-1} + \cdots + x_p t + x_{p+1}}{x_{p+2} t^q + x_{p+3} t^{q-1} + \cdots + x_{p+q+1} t + x_{p+q+2}}, \tag{8.1.2}$$

with a total of $n = p + q + 2$ *parameters.*

Rational models arise, e.g., in parameter estimation problems such as chemical reaction kinetics, signal analysis (through Laplace and z transforms), system identification and in general as a transfer function for a linear time-invariant system (similar to the above-mentioned instrument response function). In these models, the coefficients of the two polynomials or, equivalently, their roots, characterize the dynamical behavior of the system.

Rational functions are also commonly used as empirical data approximation functions. Their advantage is that they have a broader scope than polynomials, yet they are still simple to evaluate.

The basic idea of nonlinear data fitting is the same as described in Chapter 1, the only difference being that we now use a fitting model $M(\boldsymbol{x}, t)$ in which (some of) the parameters $\boldsymbol{x}$ appear nonlinearly and that leads to a nonlinear optimization problem. We are given noisy measurements $y_i = \Gamma(t_i) + e_i$ for $i = 1, \ldots, m$, where $\Gamma(t)$ is the pure-data function that gives the true relationship between t and the noise-free data, and we seek to identify the model parameters $\boldsymbol{x}$ such that $M(\boldsymbol{x}, t)$ gives the best fit to the noisy data. The likelihood function $P_{\boldsymbol{x}}$ for $\boldsymbol{x}$ is the same as before, cf. (1.3.1), leading us to consider the weighted residuals $r_i(\boldsymbol{x})/\varsigma_i = (y_i - M(\boldsymbol{x}, t_i))/\varsigma_i$ and to minimize the weighted sum-of-squares residuals.

Definition 85. ***Nonlinear least squares problem (NLLSQ).*** *In the case of white noise (where the errors have a common variance* ς^2*), find a*

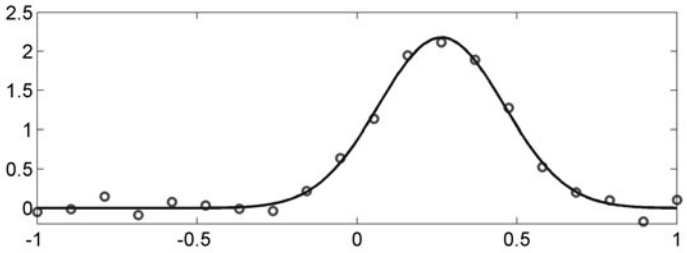

Figure 8.1.1: Fitting a non-normalized Gaussian function (full line) to noisy data (circles).

minimizer $\boldsymbol{x}^*$ *of the nonlinear objective function* f *with the special form*

$$\boxed{\min_{\boldsymbol{x}} f(\boldsymbol{x}) \equiv \min_{\boldsymbol{x}} \tfrac{1}{2} \|\boldsymbol{r}(\boldsymbol{x})\|_2^2 = \min_{\boldsymbol{x}} \tfrac{1}{2} \sum_{i=1}^{m} r_i(\boldsymbol{x})^2,} \tag{8.1.3}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ *and* $\boldsymbol{r}(\boldsymbol{x}) = (r_1(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x}))^T$ *is a vector-valued function of the residuals for each data point:*

$$r_i(\boldsymbol{x}) = y_i - M(\boldsymbol{x}, t_i), \qquad i = 1, \ldots, m,$$

in which y_i *are the measured data corresponding to* t_i *and* $M(\boldsymbol{x}, t)$ *is the model function.*

The factor $\frac{1}{2}$ is added for practical reasons as will be clear later. Simply replace $r_i(\boldsymbol{x})$ by $w_i\, r_i(\boldsymbol{x})/\varsigma_i$ in (8.1.3), where $w_i = \varsigma_i^{-1}$ is the inverse of the standard deviation for the noise in y_i, to obtain the weighted problem $\min_{\boldsymbol{x}} \frac{1}{2} \|W\boldsymbol{r}(\boldsymbol{x})\|_2^2$ for problems when the errors have not the same variance values.

Example 86. ***Fitting with the Gaussian model.*** *As an example, we consider fitting the instrument response function in Example 83 by the Gaussian model (8.1.1). First note that if we choose* $g_{\text{true}}(\tau) = \delta(\tau)$*, a delta function, then we have* $g_{\text{mes}}(t) = \Gamma(t)$*, meaning that we ideally measure the instrument response function. In practice, by sampling* $g_{\text{mes}}(t)$ *for selected values* $t_1, t_2, \ldots, t_m$ *of* t *we obtain noisy data* $y_i = \Gamma(t_i) + e_i$ *for* $i = 1, \ldots, m$ *to which we fit the Gaussian model* $M(\boldsymbol{x}, t)$*. Figure 8.1.1 illustrates this. The circles are the noisy data generated from an exact Gaussian model with* $x_1 = 2.2$*,* $x_2 = 0.26$ *and* $x_3 = 0.2$*, to which we added Gaussian noise with standard deviation* $\varsigma = 0.1$*. The full line is the least squares fit with parameters* $x_1 = 2.179$*,* $x_2 = 0.264$ *and* $x_3 = 0.194$*. In the next chapter we discuss how to compute this fit.*

Example 87. ***Nuclear magnetic resonance spectroscopy.*** *This is a technique that measures the response of nuclei of certain atoms that possess spin when they are immersed in a static magnetic field and they are exposed to a second oscillating magnetic field. NMR spectroscopy, which studies the interaction of the electromagnetic radiation with matter, is used as a non-invasive technique to obtain in vivo information about chemical changes (e.g., concentration, pH), in living organisms. An NMR spectrum of, for example, the human brain can be used to identify pathology or biochemical processes or to monitor the effect of medication.*

The model used to represent the measured NMR signals $y_i \in \mathbb{C}$ in the frequency domain is a truncated Fourier series of the Lorentzian distribution:

$$y_i \simeq M(\boldsymbol{a}, \boldsymbol{\phi}, \boldsymbol{d}, \boldsymbol{\omega}, t_i) = \sum_{k=1}^{K} a_k e^{\hat{\imath}\phi_k} e^{(-d_k + j\omega_k)t_i}, \qquad i = 1, \ldots, m,$$

where $\hat{\imath}$ denotes the imaginary unit. The parameters of the NMR signal provide information about the molecules: K represents the number of different resonance frequencies, the angular frequency ω_k of the individual spectral components identifies the molecules, the damping d_k characterizes the mobility, the amplitude a_k is proportional to the number of molecules present, and ϕ_k is the phase. All these parameters are real. To determine the NMR parameters through a least squares fit, we minimize the squared modulus of the difference between the measured spectrum and the model function:

$$\min_{\boldsymbol{a},\boldsymbol{\phi},\boldsymbol{d},\boldsymbol{\omega}} \sum_{i=1}^{m} \left| y_i - M(\boldsymbol{a}, \boldsymbol{\phi}, \boldsymbol{d}, \boldsymbol{\omega}, t_i) \right|^2 .$$

This is another example of a nonlinear least squares problem.

8.2 Unconstrained problems

Generally, nonlinear least squares problems will have multiple solutions and without a priori knowledge of the objective function it would be too expensive to determine numerically a global minimum, as one can only calculate the function and its derivatives at a limited number of points. Thus, the algorithms in this book will be limited to the determination of local minima. A certain degree of smoothness of the objective function will be required, i.e., having either one or possibly two continuous derivatives, so that the Taylor expansion applies.

Optimality conditions

Recall (Appendix C) that the gradient $\nabla f(\boldsymbol{x})$ of a scalar function $f(\boldsymbol{x})$ of n variables is the vector with elements

$$\frac{\partial f(\boldsymbol{x})}{\partial x_j}, \qquad j = 1, \ldots, n$$

and the Hessian $\nabla^2 f(\boldsymbol{x})$ is the symmetric matrix with elements

$$[\nabla^2 f(\boldsymbol{x})]_{ij} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j}, \qquad i, j = 1, \ldots, n.$$

The conditions for $\boldsymbol{x}^*$ to be a *local minimum* of a twice continouously differentiable function f are

1. **First-order necessary condition**. $\boldsymbol{x}^*$ is a stationary point, i.e., the gradient of f at $\boldsymbol{x}^*$ is zero: $\nabla f(\boldsymbol{x}^*) = 0$.
2. **Second-order sufficient conditions.** The Hessian $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite.

Now we consider the special case where f is the function in the nonlinear least squares problem (8.1.3), i.e.,

$$f(\boldsymbol{x}) = \tfrac{1}{2} \sum_{i=1}^{m} r_i(\boldsymbol{x})^2, \qquad r_i(\boldsymbol{x}) = y_i - M(\boldsymbol{x}, t_i).$$

The Jacobian $J(\boldsymbol{x})$ of the vector function $\boldsymbol{r}(\boldsymbol{x})$ is defined as the matrix with elements

$$[J(\boldsymbol{x})]_{ij} = \frac{\partial r_i(\boldsymbol{x})}{\partial x_j} = -\frac{\partial M(\boldsymbol{x}, t_i)}{\partial x_j}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Note that the ith row of $J(\boldsymbol{x})$ equals the transpose of the gradient of $r_i(\boldsymbol{x})$ and also:

$$\nabla r_i(\boldsymbol{x})^T = -\nabla M(\boldsymbol{x}, t_i)^T, \qquad i = 1, \ldots, m.$$

The elements of the gradient and the Hessian of f are given by

$$\begin{aligned} \frac{\partial f(\boldsymbol{x})}{\partial x_j} &= \sum_{i=1}^{m} r_i(\boldsymbol{x}) \frac{\partial r_i(\boldsymbol{x})}{\partial x_j}, \\ [\nabla^2 f(\boldsymbol{x})]_{k\ell} \quad = \quad \frac{\partial^2 f(\boldsymbol{x})}{\partial x_k \partial x_\ell} &= \sum_{i=1}^{m} \frac{\partial r_i(\boldsymbol{x})}{\partial x_k} \frac{\partial r_i(\boldsymbol{x})}{\partial x_\ell} + \sum_{i=1}^{m} r_i(\boldsymbol{x}) \frac{\partial^2 r_i(\boldsymbol{x})}{\partial x_k \partial x_\ell}, \end{aligned}$$

and it follows immediately that the gradient and Hessian can be written in matrix form as

$$\begin{aligned}
\nabla f(\boldsymbol{x}) &= J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}), \\
\nabla^2 f(\boldsymbol{x}) &= J(\boldsymbol{x})^T J(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x}), \\
\left[\nabla^2 r_i(\boldsymbol{x})\right]_{k\ell} &= -\left[\nabla^2 M(\boldsymbol{x}, t_i)\right]_{k\ell} = -\frac{\partial^2 M(\boldsymbol{x}, t_i)}{\partial x_k \partial x_\ell}, \quad k, \ell = 1, \ldots, m.
\end{aligned}$$

Notice the minus sign in the expression for $\nabla^2 r_i(\boldsymbol{x})$. The optimality conditions now take the special form

$$\boxed{\nabla f(\boldsymbol{x}) = J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{0}} \tag{8.2.1}$$

and

$$\boxed{\nabla^2 f(\boldsymbol{x}) = J(\boldsymbol{x})^T J(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x}) \text{ is positive definite.}} \tag{8.2.2}$$

The fact that distinguishes the least squares problem from among the general optimization problems is that the first – and often dominant – term $J(\boldsymbol{x})^T J(\boldsymbol{x})$ of the Hessian $\nabla^2 f(\boldsymbol{x})$ contains only the Jacobian $J(\boldsymbol{x})$ of $\boldsymbol{r}(\boldsymbol{x})$, i.e., only first derivatives. Observe that in the second term the second derivatives are multiplied by the residuals. Thus, if the model is adequate to represent the data, then these residuals will be small near the solution and therefore this term will be of secondary importance. In this case one gets an important part of the Hessian "for free" if one has already computed the gradient. Most specialized algorithms exploit this structural property of the Hessian.

An inspection of the Hessian of f will show two comparatively easy NLLSQ cases. As we observed above, the term $\sum_{i=1}^{m} r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x})$ will be small if the residuals are small. An additional favorable case occurs when the problem is only mildly nonlinear, i.e., all the Hessians $\nabla^2 r_i(\boldsymbol{x}) = -\nabla^2 M(\boldsymbol{x}, t_i)$ have small norm. Most algorithms that neglect this second term will then work satisfactorily. The smallest eigenvalue $\lambda_{\min}$ of $J(\boldsymbol{x})^T J(\boldsymbol{x})$ can be used to quantify the relative importance of the two terms of $\nabla^2 f(\boldsymbol{x})$: the first term of the Hessian dominates if for all $\boldsymbol{x}$ near a minimum $\boldsymbol{x}^*$ the quantities $|r_i(\boldsymbol{x})| \, \|\nabla^2 r_i(\boldsymbol{x})\|_2$ for $i = 1, \ldots, m$ are small relative to $\lambda_{\min}$. This obviously holds in the special case of a consistent problem where $\boldsymbol{r}(\boldsymbol{x}^*) = \boldsymbol{0}$.

The optimality conditions can be interpreted geometrically by observing that the gradient $\nabla f(\boldsymbol{x})$ is always a vector normal to the level set that passes through the point $\boldsymbol{x}$ (we are assuming in what follows that $f(\boldsymbol{x})$ is

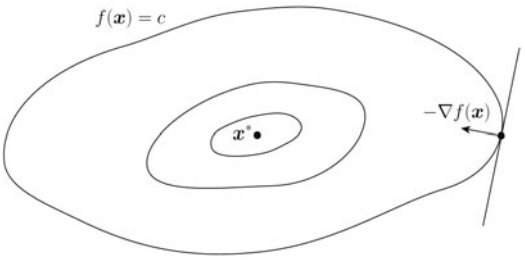

Figure 8.2.1: Level sets $L(c)$ (in $\mathbb{R}^2$ they are level curves) for a function f whose minimizer x^* is located at the black dot. The tangent plane is a line perpendicular to the gradient $\nabla f(\boldsymbol{x})$, and the negative gradient is the direction of steepest descent at $\boldsymbol{x}$.

twice differentiable, for simplicity). A *level set* $L(c)$ (level curve in $\mathbb{R}^2$) is defined formally as

$$L(c) = \{\boldsymbol{x} : \ f(\boldsymbol{x}) = c\}.$$

The tangent plane at $\boldsymbol{x}$ is $\{\, \boldsymbol{y} \in \mathbb{R}^n \mid \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) = 0\}$, which shows that $\nabla f(\boldsymbol{x})$ is its normal and thus normal to the level set at $\boldsymbol{x}$.

This leads to a geometric interpretation of directions of descent. A direction $\boldsymbol{p}$ with $\|\boldsymbol{p}\|_2 = 1$ is said to be of *descent* (from the point $\boldsymbol{x}$) if $f(\boldsymbol{x} + t\boldsymbol{p}) < f(\boldsymbol{x})$ for $0 < t < t_0$. It turns out that directions of descent can be characterized using the gradient vector. In fact, by Taylor's expansion we have

$$f(\boldsymbol{x} + t\boldsymbol{p}) = f(\boldsymbol{x}) + t\nabla f(\boldsymbol{x})^T\boldsymbol{p} + O(t^2).$$

Thus, for the descent condition to hold we need to have $\nabla f(\boldsymbol{x})^T\boldsymbol{p} < 0$, since for sufficiently small t the linear term will dominate. In addition we have $\nabla f(\boldsymbol{x})^T\boldsymbol{p} = \cos(\phi)\|\nabla f(\boldsymbol{x})\|_2$ where ϕ is the angle between $\boldsymbol{p}$ and the gradient. From this we conclude that the direction $\boldsymbol{p} = -\nabla f(\boldsymbol{x})$ is of maximum descent, or *steepest descent*, while any direction in the half-space defined by the tangent plane and containing the negative gradient is also a direction of descent, since $\cos(\phi)$ is negative there. Figure 8.2.1 illustrates this point. Note that a stationary point is one for which there are no descent directions, i.e., $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$.

We conclude with a useful geometric result when $J(\boldsymbol{x}^*)$ has full rank. In this case the characterization of a minimum can be expressed by using the so-called normal curvature matrix (see [9]) associated with the surface defined by $\boldsymbol{r}(\boldsymbol{x})$ and with respect to the normalized vector $\boldsymbol{r}(\boldsymbol{x})/\left\|\boldsymbol{r}(\boldsymbol{x})\right\|_2$, defined as

$$K(\boldsymbol{x}) = -\frac{1}{\|\boldsymbol{r}(\boldsymbol{x})\|_2}(J(\boldsymbol{x})^\dagger)^T\left(\sum_{i=1}^{m} r_i(\boldsymbol{x})\nabla^2 r_i(\boldsymbol{x})\right)J(\boldsymbol{x})^\dagger.$$

Then the condition for a minimum can be reformulated as follows.

Condition 88. *If $J(\boldsymbol{x}^*)^T(I - \|\boldsymbol{r}(\boldsymbol{x}^*)\|_2 K(\boldsymbol{x}^*))J(\boldsymbol{x}^*)$ is positive definite, then $\boldsymbol{x}^*$ is a local minimum.*

The role of local approximations

Local approximations always play an important role in the treatment of nonlinear problems and this case is no exception. We start with the Taylor expansion of the model function:

$$M(\boldsymbol{x}+\boldsymbol{h},t) = M(\boldsymbol{x},t) + \nabla M(\boldsymbol{x},t)^T\boldsymbol{h} + \tfrac{1}{2}\boldsymbol{h}^T\nabla^2 M(\boldsymbol{x},t)\boldsymbol{h} + O(\|\boldsymbol{h}\|_2^3).$$

Now consider the ith residual $r_i(\boldsymbol{x}) = y_i - M(\boldsymbol{x},t_i)$, whose Taylor expansion for $i = 1,\ldots,m$ is given by

$$\begin{aligned} r_i(\boldsymbol{x}+\boldsymbol{h}) &= y_i - M(\boldsymbol{x}+\boldsymbol{h},t_i) \\ &= y_i - M(\boldsymbol{x},t_i) - \nabla M(\boldsymbol{x},t_i)^T\boldsymbol{h} - \\ &\quad \tfrac{1}{2}\boldsymbol{h}^T\nabla^2 M(\boldsymbol{x},t_i)\boldsymbol{h} + O(\|\boldsymbol{h}\|_2^3). \end{aligned}$$

Hence we can write

$$\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h}) = \boldsymbol{r}(\boldsymbol{x}) + J(\boldsymbol{x})\,\boldsymbol{h} + O(\|\boldsymbol{h}\|_2^2),$$

which is a local linear approximation at $\boldsymbol{x}$ valid for small $\boldsymbol{h}$. If we keep $\boldsymbol{x}$ fixed and consider the minimization problem

$$\min_{\boldsymbol{h}} \|\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h})\|_2 \simeq \min_{\boldsymbol{h}} \|J(\boldsymbol{x})\,\boldsymbol{h} + \boldsymbol{r}(\boldsymbol{x})\|_2, \tag{8.2.3}$$

it is clear that we can approximate locally the nonlinear least squares problem with a linear one in the variable $\boldsymbol{h}$. Moreover, we see that the $\boldsymbol{h}$ that minimizes $\|\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h})\|_2$ can be approximated by

$$\boldsymbol{h} \approx -(J(\boldsymbol{x})^T J(\boldsymbol{x}))^{-1} J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) = -J(\boldsymbol{x})^\dagger \boldsymbol{r}(\boldsymbol{x}),$$

where we have neglected higher-order terms. In the next chapter we shall see how this expression provides a basis for some of the numerical methods for solving the NLLSQ problem.

As a special case, let us now consider the local approximation (8.2.3) at a least squares solution $\boldsymbol{x}^*$ (a local minimizer) where the local, linear least squares problem takes the form

$$\min_{\boldsymbol{h}} \|J(\boldsymbol{x}^*)\,\boldsymbol{h} + \boldsymbol{r}(\boldsymbol{x}^*)\|_2.$$

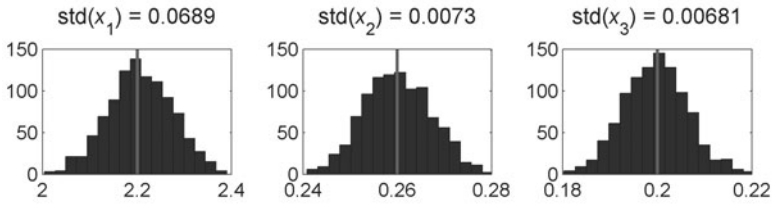

Figure 8.2.2: Histograms and standard deviations for the least squares solutions to 1000 noisy realizations of the Gaussian model fitting problem. The vertical lines show the exact values of the parameters.

If we introduce $\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{h}$, then we can reformulate the above problem as one in $\boldsymbol{x}$,

$$\min_{\boldsymbol{x}} \|J(\boldsymbol{x}^*)\,\boldsymbol{x} - J(\boldsymbol{x}^*)\,\boldsymbol{x}^* + \boldsymbol{r}(\boldsymbol{x}^*)\|_2,$$

which leads to an approximation of the covariance matrix for $\boldsymbol{x}^*$ by using the covariance matrix for the above linear least squares problem, i.e.,

$$\begin{aligned} \mathrm{Cov}(\boldsymbol{x}^*) &\simeq J(\boldsymbol{x}^*)^\dagger \mathrm{Cov}\big(J(\boldsymbol{x}^*)\,\boldsymbol{x}^* - \boldsymbol{r}(\boldsymbol{x}^*)\big)(J(\boldsymbol{x}^*)^\dagger)^T \\ &= J(\boldsymbol{x}^*)^\dagger \mathrm{Cov}\big(\boldsymbol{r}(\boldsymbol{x}^*) - J(\boldsymbol{x}^*)\,\boldsymbol{x}^*\big)(J(\boldsymbol{x}^*)^\dagger)^T \\ &= J(\boldsymbol{x}^*)^\dagger \mathrm{Cov}(\boldsymbol{y})(J(\boldsymbol{x}^*)^\dagger)^T. \end{aligned} \tag{8.2.4}$$

Here, we have used that $\boldsymbol{r}(\boldsymbol{x}^*) = \boldsymbol{y} - (M(\boldsymbol{x}^*, t_1), \ldots, M(\boldsymbol{x}^*, t_m))^T$ and that all the terms except $\boldsymbol{y}$ in $\boldsymbol{r}(\boldsymbol{x}^*) + J(\boldsymbol{x}^*)\,\boldsymbol{x}^*$ are considered constant.

The above equation provides a way to approximately assess the uncertainties in the least squares solution $\boldsymbol{x}^*$ for the nonlinear case, similar to the result in Section 2.1 for the linear case. In particular, we see that the Jacobian $J(\boldsymbol{x}^*)$ at the solution plays the role of the matrix in the linear case. If the errors $\boldsymbol{e}$ in the data are uncorrelated with the exact data and have covariance $\mathrm{Cov}(\boldsymbol{e}) = \varsigma^2 I$, then $\mathrm{Cov}(\boldsymbol{x}^*) \simeq \varsigma^2 \big(J(\boldsymbol{x}^*)^T J(\boldsymbol{x}^*)\big)^{-1}$.

Example 89. *To illustrate the use of the above covariance matrix estimate, we return to the Gaussian model $M(\boldsymbol{x}, t)$ from Examples 83 and 86, with exact parameters $x_1 = 2.2$, $x_2 = 0.26$ and $x_3 = 0.2$ and with noise level $\varsigma = 0.1$. The elements of the Jacobian for this problem are, for $i = 1, \ldots, m$:*

$$\begin{aligned} [J(\boldsymbol{x})]_{i,1} &= e^{-(t_i - x_2)^2/(2x_3^2)}, \\ [J(\boldsymbol{x})]_{i,2} &= x_1 \frac{t_i - x_2}{x_3^2}\, e^{-(t_i - x_2)^2/(2x_3^2)}, \\ [J(\boldsymbol{x})]_{i,3} &= x_1 \frac{(t_i - x_2)^2}{x_3^3}\, e^{-(t_i - x_2)^2/(2x_3^2)}. \end{aligned}$$

The approximate standard deviations for the three estimated parameters are the square roots of the diagonal of $\varsigma^2\big(J(\boldsymbol{x}^*)^T J(\boldsymbol{x}^*)\big)^{-1}$. *In this case we get the three values*

$$6.58 \cdot 10^{-2}, \qquad 6.82 \cdot 10^{-3}, \qquad 6.82 \cdot 10^{-3}.$$

We compute the least squares solution $\boldsymbol{x}^*$ *for 1000 realizations of the noise. Histograms of these parameters are shown in Figure 8.2.2. The standard deviations of these estimated parameters are*

$$\text{std}(x_1) = 6.89 \cdot 10^{-2}, \quad \text{std}(x_2) = 7.30 \cdot 10^{-3}, \quad \text{std}(x_3) = 6.81 \cdot 10^{-3}.$$

In this example, the theoretical standard deviation estimates are in very good agreement with the observed ones.

8.3 Optimality conditions for constrained problems

In the previous section we reviewed the conditions for a point $\boldsymbol{x}^*$ to be a local minimizer of an unconstrained nonlinear least squares problem. Now we consider a constrained problem

$$\boxed{\min_{\boldsymbol{x} \in C} \tfrac{1}{2} \left\| \boldsymbol{r}(\boldsymbol{x}) \right\|_2^2,} \tag{8.3.1}$$

where the residual vector $\boldsymbol{r}(\boldsymbol{x})$ is as above and the set $C \subset \mathbb{R}^n$ (called the feasible region) is defined by a set of inequality constraints:

$$C = \{\, \boldsymbol{x} \,|\, c_i(\boldsymbol{x}) \leq 0,\ i = 1, \ldots, p\},$$

where the functions $c_i(\boldsymbol{x})$ are twice differentiable. In the unconstrained case, it could be that the only minimum is at infinity (think of $f(x) = a + bx$). If instead we limit the variables to be in a closed, bounded set, then the Bolzano-Weierstrass theorem [79] ensures us that there will be at least one minimum point.

A simple way to define a bounded set C is to impose constraints on the variables. The simplest constraints are those that set bounds on the variables: $l_i \leq x_i \leq u_i$ for $i = 1, \ldots, p$. A full set of such constraints (i.e., $p = n$) defines a bounded box in $\mathbb{R}^n$ and then we are guaranteed that $f(\boldsymbol{x})$ will have a minimum in the set. For general constraints some of the functions $c_i(\boldsymbol{x})$ can be nonlinear.

The points that satisfy all the constraints are called *feasible.* A constraint is *active* if it is satisfied with equality, i.e., the point is on a boundary of the feasible set. If we ignore the constraints, unconstrained minima can

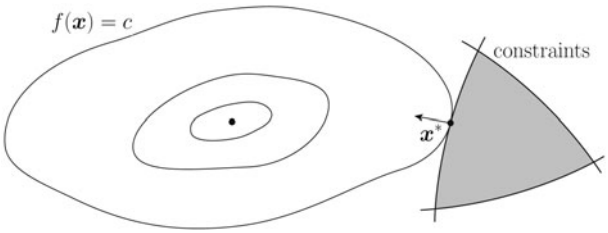

Figure 8.3.1: Optimality condition for constrained optimization. The shaded area illustrates the feasible region C defined by three constraints. One of the constraints is active at the solution $\boldsymbol{x}^*$ and at this point the gradient of f is collinear with the normal to the constraint.

occur inside or outside the feasible region. If they occur inside, then the optimality conditions are the same as in the unconstrained case. However, if all unconstrained minima are infeasible, then a constrained minimum must occur on the boundary of the feasible region and the optimality conditions will be different.

Let us think geometrically (better still, in $\mathbb{R}^2$, see Figure 8.3.1). The level curve values of $f(\boldsymbol{x})$ decrease toward the unconstrained minimum. Thus, a constrained minimum will lie on the level curve with the lowest value of $f(\boldsymbol{x})$ that is still in contact with the feasible region and at the constrained minimum there should not be any feasible descent directions. A moment of reflection tells us that the level curve should be tangent to the active constraint (the situation is more complicated if more than one constraint is active). That means that the normal to the active constraint and the gradient (pointing to the outside of the constraint region) at that point are collinear, which can be expressed as

$$\nabla f(\boldsymbol{x}) + \lambda \nabla c_i(\boldsymbol{x}) = 0, \tag{8.3.2}$$

where $c_i(\boldsymbol{x})$ represents the active constraint and $\lambda > 0$ is a so-called Lagrange multiplier for the constraint. (If the constraint were $c_i(\boldsymbol{x}) \geq 0$, then we should require $\lambda < 0$.) We see that (8.3.2) is true because, if $\boldsymbol{p}$ is a descent direction, we have

$$\boldsymbol{p}^T \nabla f(\boldsymbol{x}) + \lambda \boldsymbol{p}^T \nabla c_i(\boldsymbol{x}) = 0$$

or, since $\lambda > 0$,

$$\boldsymbol{p}^T \nabla c_i(\boldsymbol{x}) > 0,$$

and therefore $\boldsymbol{p}$ must point outside the feasible region; see Figure 8.3.1 for an illustration. This is a simplified version of the famous Karush-Kuhn-Tucker

first-order optimality conditions for general nonlinear optimization [139, 147]. Curiously enough, in the original Kuhn-Tucker paper these conditions are derived from considerations based on multiobjective optimization!

The general case when more than one constraint is active at the solution, can be discussed similarly. Now the infeasible directions instead of being in a half-space determined by the tangent plane of a single constraint will be in a cone formed by the normals to the tangent planes associated with the various active constraints. The corresponding optimality condition is

$$\boxed{\nabla f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i \nabla c_i(\boldsymbol{x}) = 0,} \tag{8.3.3}$$

where the Lagrange multipliers satisfy $\lambda_i > 0$ for the active constraints and $\lambda_i = 0$ for the inactive ones. We refer to [170] for more details about optimality constraints and Lagrange multipliers.

8.4 Separable nonlinear least squares problems

In many practical applications the unknown parameters of the NLLSQ problem can be separated into two disjoint sets, so that the optimization with respect to one set is easier than with respect to the other. This suggests the idea of eliminating the parameters in the easy set and minimizing the resulting functional, which will then depend only on the remaining variables. A natural situation that will be considered in detail here arises when some of the variables appear linearly.

The initial approach to this problem was considered by H. Scolnik in his Doctoral Thesis at the University of Zurich. A first paper with generalizations was [113]. This was followed by an extensive generalization that included a detailed algorithmic description [101, 102] and a computer program called VARPRO that become very popular and is still in use. For a recent detailed survey of applications see [103] and [198]. A paper that includes constraints is [142]. For multiple right hand sides see [98, 141].

A separable nonlinear least squares problem has a model of the special form

$$\boxed{M(\boldsymbol{a}, \boldsymbol{\alpha}, t) = \sum_{j=1}^{n} a_j \, \phi_j(\boldsymbol{\alpha}, t),} \tag{8.4.1}$$

where the two vectors $\boldsymbol{a} \in \mathbb{R}^n$ and $\boldsymbol{\alpha} \in \mathbb{R}^k$ contain the parameters to be determined and $\phi_j(\boldsymbol{\alpha}, t)$ are functions in which the parameters $\boldsymbol{\alpha}$ appear nonlinearly. Fitting this model to a set of m data points (t_i, y_i) with $m > n + k$ leads to the least squares problem

$$\min_{\mathbf{a},\boldsymbol{\alpha}} f(\boldsymbol{a},\, \boldsymbol{\alpha}) = \min_{\mathbf{a},\boldsymbol{\alpha}} \tfrac{1}{2}\|\boldsymbol{r}(\boldsymbol{a},\, \boldsymbol{\alpha})\|_2^2 = \min_{\mathbf{a},\boldsymbol{\alpha}} \tfrac{1}{2}\|\boldsymbol{y} - \Phi(\boldsymbol{\alpha})\, \boldsymbol{a}\|_2^2.$$

In this expression, $\Phi(\boldsymbol{\alpha})$ is an $m \times n$ matrix function with elements

$$\Phi(\boldsymbol{\alpha}) = \phi_j(\boldsymbol{\alpha},\, t_i), \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

The special case when $k = n$ and $\phi_j = e^{\alpha_j t}$ is called *exponential data fitting*. In this as in other separable nonlinear least squares problems the matrix $\Phi(\boldsymbol{\alpha})$ is often ill-conditioned.

The variable projection algorithm – to be discussed in detail in the next chapter – is based on the following ideas. For any fixed $\boldsymbol{\alpha}$, the problem

$$\min_{\mathbf{a}} \tfrac{1}{2}\|\boldsymbol{y} - \Phi(\boldsymbol{\alpha})\, \boldsymbol{a}\|_2^2$$

is linear with minimum-norm solution $\boldsymbol{a}^* = \Phi(\boldsymbol{\alpha})^\dagger \boldsymbol{y}$, where $\Phi(\boldsymbol{\alpha})^\dagger$ is the Moore-Penrose generalized inverse of the rectangular matrix $\Phi(\boldsymbol{\alpha})$, as we saw in Chapter 3.

Substituting this value of $\boldsymbol{a}$ into the original problem, we obtain a reduced nonlinear problem depending only on the nonlinear parameters $\boldsymbol{\alpha}$, with the associated function

$$f_2(\boldsymbol{\alpha}) = \tfrac{1}{2}\|\boldsymbol{y} - \Phi(\boldsymbol{\alpha})\Phi(\boldsymbol{\alpha})^\dagger\, \boldsymbol{y}\|_2^2 = \tfrac{1}{2}\|P_{\Phi(\boldsymbol{\alpha})}\, \boldsymbol{y}\|_2^2,$$

where, for a given $\boldsymbol{\alpha}$, the matrix

$$P_{\Phi(\boldsymbol{\alpha})} = I - \Phi(\boldsymbol{\alpha})\Phi(\boldsymbol{\alpha})^\dagger$$

is the projector onto the orthogonal complement of the column space of the matrix $\Phi(\boldsymbol{\alpha})$. Thus the name variable projection given to this reduction procedure. The solution of the nonlinear least squares problem $\min_{\boldsymbol{\alpha}} f_2(\boldsymbol{\alpha})$ is discussed in the next chapter.

Once a minimizer $\boldsymbol{\alpha}^*$ for $f_2(\boldsymbol{\alpha})$ is obtained, it is substituted into the linear problem $\min_{\mathbf{a}} \tfrac{1}{2}\|\boldsymbol{y} - \Phi(\boldsymbol{\alpha}^*)\, \boldsymbol{a}\|_2^2$, which is then solved to obtain $\boldsymbol{a}^*$. The least squares solution to the original problem is then $(\boldsymbol{a}^*, \boldsymbol{\alpha}^*)$.

The justification for the variable projection algorithm is the following theorem (Theorem 2.1 proved in [101]), which essentially states that the set of stationary points of the original functional and that of the reduced one are identical.

Theorem 90. *Let $f(\boldsymbol{a},\, \boldsymbol{\alpha})$ and $f_2(\boldsymbol{\alpha})$ be defined as above. We assume that in an open set Ω containing the solution, the matrix $\Phi(\boldsymbol{\alpha})$ has constant rank $r \leqq \min(m, n)$.*

1. *If $\boldsymbol{\alpha}^*$ is a critical point (or a global minimizer in Ω) of $f_2(\boldsymbol{\alpha})$ and $\boldsymbol{a}^* = \Phi(\boldsymbol{\alpha})^\dagger \boldsymbol{y}$, then $(\boldsymbol{a}^*, \boldsymbol{\alpha}^*)$ is a critical point of $f(\boldsymbol{a},\, \boldsymbol{\alpha})$ (or a global minimizer in Ω) and $f(\boldsymbol{a}^*,\, \boldsymbol{\alpha}^*) = f_2(\boldsymbol{\alpha}^*)$.*

2. *If* $(\boldsymbol{a}^*, \boldsymbol{\alpha}^*)$ *is a global minimizer of* $f(\boldsymbol{a}, \boldsymbol{\alpha})$ *for* $\boldsymbol{\alpha} \in \Omega$*, then* $\boldsymbol{\alpha}^*$ *is a global minimizer of* $f_2(\boldsymbol{\alpha})$ *in* Ω *and* $f_2(\boldsymbol{\alpha}^*) = f(\boldsymbol{a}^*, \boldsymbol{\alpha}^*)$*. Furthermore, if there is a unique* $\boldsymbol{a}^*$ *among the minimizing pairs of* $f(\boldsymbol{a}, \boldsymbol{\alpha})$ *then* $\boldsymbol{a}^*$ *must satisfy* $\boldsymbol{a}^* = \Phi(\boldsymbol{\alpha}^*)^\dagger \boldsymbol{y}$*.*

8.5 Multiobjective optimization

This is a subject that is seldom treated in optimization books, although it arises frequently in financial optimization, decision and game theory and other areas. In recent times it has increasingly been recognized that many engineering problems are also of this kind. We have also already mentioned that most regularization procedures are actually bi-objective problems (see [240, 241] for an interesting theory and algorithm).

The basic unconstrained problem is

$$\min_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}),$$

where now $\boldsymbol{f} \in \mathbb{R}^k$ is a vector function. Such problems arise in areas of interest of this book, for instance, in cooperative fitting and inversion, when measurements of several physical processes on the same sample are used to determine properties that are interconnected.

In general, it will be unlikely that the k objective functions share a common minimum point, so the theory of multiobjective optimization is different from the single-objective optimization case. The objectives are usually in conflict and a compromise must be chosen. The optimality concept is here the so-called Pareto equilibrium: *a point* $\boldsymbol{x}$ *is a Pareto equilibrium point if there is no other point for which all functions have smaller values.*

This notion can be global (as stated) or local if it is restricted to a neighborhood of $\boldsymbol{x}$. In general there will be infinitely many such points. In the absence of additional information, each Pareto point would be equally acceptable. Let us now see if we can give a more geometric interpretation of this condition, based on some of the familiar concepts used earlier.

For simplicity let us consider the case $k = 2$. First of all we observe that the individual minimum points

$$\boldsymbol{x}_i^* = \arg\min_{\boldsymbol{x}} f_i(\boldsymbol{x}), \qquad i = 1, 2$$

are Pareto optimal. We consider the level sets corresponding to the two functions and observe that at a point at which these two level sets are tangent and their corresponding normals are in opposite directions, there will be no common directions of descent for both functions. But that means that there is no direction in which we can move so that both functions are improved, i.e., this is a Pareto equilibrium point. Analytically this is expressed as

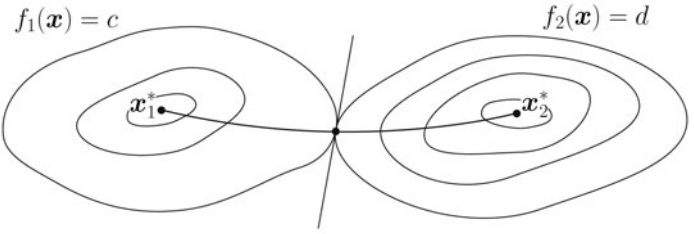

Figure 8.5.1: Illustration of a multiobjective optimization problem. The two functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ have minimizers $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$. The curve between these two points is the set of Pareto points.

$$\frac{\nabla f_1(\boldsymbol{x})}{\|\nabla f_1(\boldsymbol{x})\|_2} + \frac{\nabla f_2(\boldsymbol{x})}{\|\nabla f_2(\boldsymbol{x})\|_2} = \mathbf{0},$$

which can be rewritten as

$$\lambda \nabla f_1(\boldsymbol{x}) + (1-\lambda)\nabla f_2(\boldsymbol{x}) = \mathbf{0}, \qquad 0 \le \lambda \le 1.$$

It can be shown that for convex functions, all the Pareto points are parametrized by this aggregated formula, which coincides with the optimality condition for the scalar optimization problems

$$\min_{\boldsymbol{x}} \left[\lambda f_1(\boldsymbol{x}) + (1-\lambda) f_2(\boldsymbol{x})\right], \qquad 0 \le \lambda \le 1.$$

This furnishes a way to obtain the Pareto points by solving a number of scalar optimization problems. Figure 8.5.1 illustrates this; the curve between the two minimizers $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$ is the set of Pareto points.

Curiously enough, these optimality conditions were already derived in the seminal paper on nonlinear programming by Kuhn and Tucker in 1951 [147].

A useful tool is given by the graph in phase space of the Pareto points, the so-called Pareto front (see 8.5.2.) In fact, this graph gives a complete picture of all possible solutions, and it is then usually straightforward to make a decision by choosing an appropriate trade-off between the objectives. Since one will usually be able to calculate only a limited number of Pareto points, it is important that the Pareto front be uniformly sampled. In [190, 194], a method based on continuation in λ with added distance constraints produces automatically equispaced representations of the Pareto front. In Figure 8.5.2 we show a uniformly sampled Pareto front for a bi-objective problem.

The ϵ-constraint method of Haimes [114] is frequently used to solve this type of problem in the case that a hierarchical order of the objectives is

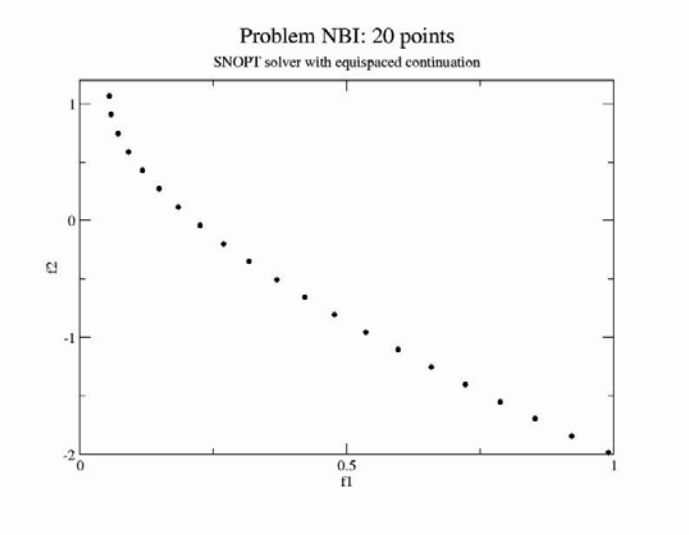

Figure 8.5.2: Evenly spaced Pareto front (in (f_1, f_2) space) for a bi-objective problem [190].

known and one can make an a priori call on a compromise upper value for the secondary objective. It transforms the bi-objective problem into a single-objective constrained minimization of the main goal. The constraint is the upper bound of the second objective. In other words, one minimizes the first objective subject to an acceptable level in the second objective (that better be larger than the minimum). From what we saw before, the resulting solution may be sub-optimal.

A good reference for the theoretical and practical aspects of nonlinear multiobjective optimization is [161].

Chapter 9

Algorithms for Solving Nonlinear LSQ Problems

The classical method of Newton and its variants can be used to solve the nonlinear least squares problem formulated in the previous chapter. Newton's method for optimization is based on a second-order Taylor approximation of the objective function $f(\boldsymbol{x})$ and subsequent minimization of the resulting approximate function. Alternatively, one can apply Newton's method to solve the nonlinear system of equations $\nabla f(\boldsymbol{x}) = \mathbf{0}$.

The Gauss-Newton method is a simplification of the latter approach for problems that are almost consistent or mildly nonlinear, and for which the second term in the Hessian $\nabla^2 f(\boldsymbol{x})$ can be safely neglected. The Levenberg-Marquardt method can be considered a variant of the Gauss-Newton method in which stabilization (in the form of Tikhonov regularization, cf. Section 10) is applied to the linearized steps in order to solve problems with ill-conditioned or rank-deficient Jacobian $J(\boldsymbol{x})$.

Due to the local character of the Taylor approximation one can only obtain local convergence, in general. In order to get global convergence, the methods need to be combined with a line search. Global convergence means convergence to a local minimum from any initial point. For convergence to a global minimum one needs to resort, for instance, to a Monte Carlo technique that provides multiple random initial points, or to other costlier methods [191].

We will describe first the different methods and then consider the common difficulties, such as how to start and end and how to ensure descent at each step.

9.1 Newton's method

If we assume that $f(\boldsymbol{x})$ is twice continuously differentiable, then we can use Newton's method to solve the system of nonlinear equations

$$\nabla f(\boldsymbol{x}) = J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{0},$$

which provides local stationary points for $f(\boldsymbol{x})$. Written in terms of derivatives of $\boldsymbol{r}(\boldsymbol{x})$ and starting from an initial guess $\boldsymbol{x}_0$ this version of the Newton iteration takes the form

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \left(\nabla^2 f(\boldsymbol{x}_k)\right)^{-1} \nabla f(\boldsymbol{x}_k) \\ &= \boldsymbol{x}_k - \left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + S(\boldsymbol{x}_k)\right)^{-1} J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k), \\ & \quad k = 0, 1, 2, \ldots \end{aligned}$$

where $S(\boldsymbol{x}_k)$ denotes the matrix

$$S(\boldsymbol{x}_k) = \sum_{i=1}^{m} r_i(\boldsymbol{x}_k) \nabla^2 r_i(\boldsymbol{x}_k). \tag{9.1.1}$$

As usual, no inverse is calculated to obtain a new iterate, but rather a linear system is solved by a direct or iterative method to obtain the correction $\Delta \boldsymbol{x}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$

$$\left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + S(\boldsymbol{x}_k)\right) \Delta \boldsymbol{x}_k = -J^T(\boldsymbol{x}_k) \boldsymbol{r}(\boldsymbol{x}_k). \tag{9.1.2}$$

The method is locally quadratically convergent as long as $\nabla^2 f(\boldsymbol{x})$ is Lipschitz continuous and positive definite in a neighborhood of $\boldsymbol{x}^*$. This follows from a simple adaptation of the Newton convergence theorem in [66], which leads to the result

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|_2 \leq \gamma \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2^2, \qquad k = 0, 1, 2, \ldots .$$

The constant γ is a measure of the nonlinearity of $\nabla f(\boldsymbol{x})$, it depends on the Lipschitz constant for $\nabla^2 f(\boldsymbol{x})$ and a bound for $\left\|\nabla^2 f(\mathbf{x}^*)^{-1}\right\|_2^2$, the size of the residual does not appear. The convergence rate will depend on the nonlinearity, but the convergence itself will not. The foregoing result implies that Newton's method is usually very fast in the final stages, close to the solution.

In practical applications Newton's iteration may not be performed exactly and theorems 4.22 and 4.30 in [143, 183] give convergence results when the errors are controlled appropriately.

Note that the Newton correction $\Delta \boldsymbol{x}_k$ will be a descent direction (as explained in the previous chapter), as long as $\nabla^2 f(\boldsymbol{x}_k) = J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) +$

$S(\boldsymbol{x}_k)$ is positive definite. In fact, using the definition of positive definite matrices one obtains, by multiplying both sides of (9.1.2)

$$0 < \Delta\boldsymbol{x}_k^T \left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + S(\boldsymbol{x}_k)\right) \Delta\boldsymbol{x}_k = -\Delta\boldsymbol{x}_k^T J^T(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k).$$

Therefore, the correction $\Delta\boldsymbol{x}_k$ is in the same half-space as the steepest descent direction $-J^T(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k)$. Although the Newton correction is a descent direction, the step size may be too large, since the linear model is only locally valid. Therefore to ensure convergence, Newton's method is used with step-length control to produce a more robust algorithm.

One of the reasons why Newton's method is not used more frequently for nonlinear least squares problem is that its good convergence properties come at a price: the mn^2 derivatives appearing in $S(\boldsymbol{x}_k)$ must be computed! This can be expensive and often the derivatives are not even available and thus must be substituted by finite differences. Special cases where Newton's method is a good option are when $S(\boldsymbol{x}_k)$ is sparse, which happens frequently if $J(\boldsymbol{x}_k)$ is sparse or when $S(\boldsymbol{x}_k)$ involves exponentials or trigonometric functions that are easy to compute. Also, if one has access to the code that calculates the model, automatic differentiation can be used [111].

If the second derivatives term in $\nabla^2 f(\boldsymbol{x}_k) = J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + S(\boldsymbol{x}_k)$ is unavailable or too expensive to compute and hence approximated, the resulting algorithm is called a quasi-Newton method and although the convergence will no longer be quadratic, superlinear convergence can be attained. It is important to point out that only the second term of $\nabla^2 f(\boldsymbol{x}_k)$ needs to be approximated since the first term $J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)$ has already been computed. A successful strategy is to approximate $S(\boldsymbol{x}_k)$ by a secant-type term, using updated gradient evaluations:

$$S(\boldsymbol{x}_k) \simeq \sum_{i=1}^{m} r_i(\boldsymbol{x}_k)\widetilde{G}_i(\boldsymbol{x}_k),$$

where the Hessian terms $\nabla^2 r_i(\boldsymbol{x}_k)$ are approximated from the condition

$$\widetilde{G}_i(\boldsymbol{x}_k)\,(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) = \nabla r_i(\boldsymbol{x}_k) - \nabla r_i(\boldsymbol{x}_{k-1}).$$

This condition would determine a secant approximation for $n = 1$, but in the higher-dimensional case it must be complemented with additional requirements on the matrix $\widetilde{G}_i(\boldsymbol{x}_k)$: it must be symmetric and it must satisfy the so-called least-change secant condition, i.e., that it be most similar to the approximation $\widetilde{G}_i(\boldsymbol{x}_{k-1})$ in the previous step. In [67], local superlinear convergence is proved, under the assumptions of Lipschitz continuity and bounded inverse of $\nabla^2 f(\boldsymbol{x})$.

9.2 The Gauss-Newton method

If the problem is only mildly nonlinear or if the residual at the solution (and therefore in a reasonable-sized neighborhood) is small, a good alternative to Newton's method is to neglect the second term $S(\boldsymbol{x}_k)$ of the Hessian altogether. The resulting method is referred to as the Gauss-Newton method, where the computation of the step $\Delta\boldsymbol{x}_k$ involves the solution of the linear system

$$\left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)\right) \Delta\boldsymbol{x}_k = -J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k) \tag{9.2.1}$$

and $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x}_k$.

Note that in the full-rank case these are actually the normal equations for the linear least squares problem

$$\min_{\Delta\boldsymbol{x}_{k+1}} \|J(\boldsymbol{x}_k)\,\Delta\boldsymbol{x}_k - (-\boldsymbol{r}(\boldsymbol{x}_k))\|_2 \tag{9.2.2}$$

and thus $\Delta\boldsymbol{x}_k = -J(\boldsymbol{x}_k)^\dagger \boldsymbol{r}(\boldsymbol{x}_k)$. By the same argument as used for the Newton method, this is a descent direction if $J(\boldsymbol{x}_k)$ has full rank.

We note that the linear least squares problem in (9.2.2), which defines the Gauss-Newton direction $\Delta\boldsymbol{x}_k$, can also be derived from the principle of local approximation discussed in the previous chapter. When we linearize the residual vector $\boldsymbol{r}(\boldsymbol{x}_k)$ in the kth step we obtain the approximation in (8.2.3) with $\boldsymbol{x} = \boldsymbol{x}_k$, which is identical to (9.2.2).

The convergence properties of the Gauss-Newton method can be summarized as follows [66] (see also [184] for an early proof of convergence):

Theorem 91. *Assume that*

- *$f(\boldsymbol{x})$ is twice continuously differentiable in an open convex set $\mathcal{D}$.*
- *$J(\boldsymbol{x})$ is Lipschitz continuous with constant γ, and $\|J(\boldsymbol{x})\|_2 \leqslant \alpha$.*
- *There is a stationary point $\boldsymbol{x}^* \in \mathcal{D}$, and*
- *for all $\boldsymbol{x} \in \mathcal{D}$ there exists a constant σ such that*

$$\left\|(J(\boldsymbol{x}) - J(\boldsymbol{x}^*))^T \boldsymbol{r}(\boldsymbol{x}^*)\right\|_2 \leqslant \sigma \|\boldsymbol{x} - \boldsymbol{x}^*\|_2.$$

If σ is smaller than λ, the smallest eigenvalue of $J(\boldsymbol{x}^*)^T J(\boldsymbol{x}^*)$, then for any $c \in (1, \lambda/\sigma)$ there exists a neighborhood so that the Gauss-Newton sequence converges linearly to $\boldsymbol{x}^*$ starting from any initial point $\boldsymbol{x}_0$ in $\mathcal{D}$:

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|_2 \leqslant \frac{c\sigma}{\lambda} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 + \frac{c\alpha\gamma}{2\lambda} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2^2$$

and

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|_2 \leqslant \frac{c\sigma + \lambda}{2\lambda} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 < \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 .$$

If the problem is consistent, i.e., $\boldsymbol{r}(\boldsymbol{x}^*) = \mathbf{0}$*, there exists a (maybe smaller) neighborhood where the convergence will be quadratic.*

The Gauss-Newton method can produce the whole spectrum of convergence: if $S(\boldsymbol{x}_k) = 0$, or is very small, the convergence can be quadratic, but if $S(\boldsymbol{x}_k)$ is too large there may not even be local convergence. The important constant is σ, which can be considered as an approximation of $\|S(\boldsymbol{x}^*)\|_2$ since, for $\boldsymbol{x}$ sufficiently close to $\boldsymbol{x}^*$, it can be shown that

$$\left\|(J(\boldsymbol{x}) - J(\boldsymbol{x}^*))^T \boldsymbol{r}(\boldsymbol{x}^*)\right\|_2 \simeq \|S(\boldsymbol{x}^*)\|_2 \, \|\boldsymbol{x} - \boldsymbol{x}^*\|_2 \, .$$

The ratio σ/λ must be less than 1 for convergence. The rate of convergence is inversely proportional to the "size" of the nonlinearity or the residual, i.e., the larger $\|S(\boldsymbol{x}_k)\|_2$ is in comparison to $\|J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)\|_2$, the slower the convergence.

As we saw above, the Gauss-Newton method produces a direction $\Delta\boldsymbol{x}_k$ that is of descent, but due to the local character of the underlying approximation, the step length may be incorrect, i.e., $f(x_k + \Delta\boldsymbol{x}_k) > f(\boldsymbol{x}_k)$. This suggests a way to improve the algorithm, namely, to use damping, which is a simple mechanism that controls the step length to ensure a sufficient reduction of the function. An alternative is to use a trust-region strategy to define the direction; this leads to the Levenberg-Marquardt algorithm discussed later on.

Algorithm damped Gauss-Newton

- Start with an initial guess $\boldsymbol{x}_0$ and iterate for $k = 0, 1, 2, \ldots$
- Solve $\min_{\Delta\boldsymbol{x}_k} \|J(\boldsymbol{x}_k)\, \Delta\boldsymbol{x}_k + \boldsymbol{r}(\boldsymbol{x}_k)\|_2$ to compute the correction $\Delta\boldsymbol{x}_k$.
- Choose a step length α_k so that there is enough descent.
- Calculate the new iterate: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \Delta\boldsymbol{x}_k$.
- Check for convergence.

Several methods for choosing the step-length parameter α_k have been proposed and the key idea is to ensure descent by the correction step $\alpha_k \Delta\boldsymbol{x}_k$. One popular choice is the Armijo condition (see Section 9.4), which uses a constant $\alpha_k \in (0, 1)$ to ensure enough descent in the value of the objective:

$$\begin{aligned} f(\boldsymbol{x}_k + \alpha_k \Delta\boldsymbol{x}_k) &< f(\boldsymbol{x}_k) + \alpha_k \nabla f(\boldsymbol{x}_k)^T \Delta\boldsymbol{x}_k \\ &= f(\boldsymbol{x}_k) + \alpha_k \boldsymbol{r}(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)^T \Delta\boldsymbol{x}_k. \end{aligned} \tag{9.2.3}$$

This condition ensures that the reduction is (at least) proportional to both the parameter α_k and the directional derivative $\nabla f(\boldsymbol{x}_k)^T \Delta\boldsymbol{x}_k$.

Using the two properties above, a descent direction and an appropriate step length, the damped Gauss-Newton method is locally convergent

y	x^*	$f(x^*)$	$J(x^*)^T J(x^*)$	$S(x^*)$
8	0.6932	0	644.00	0
3	0.4401	1.639	151.83	9.0527
-1	0.0447	6.977	17.6492	8.3527
-8	-0.7915	41.145	0.4520	2.9605

Table 9.1: Data for the test problem for four values of the scalar y. Note that x^*, $J(x^*)$ and $S(x^*)$ are scalars in this example.

and often globally convergent. Still, the convergence rate may be slow for problems for which the standard algorithm is inadequate. Also, the inefficiency of the Gauss-Newton method when applied to problems with ill-conditioned or rank-deficient Jacobian has not been addressed; the next algorithm, Levenberg-Marquardt, will consider this issue.

Example 92. *The following example from [66] will clarify the above convergence results. We fit the one-parameter model $M(x,t) = e^{xt}$ to the data set*

$$(t_1, y_1) = (1, 2), \qquad (t_2, y_2) = (2, 4), \qquad (t_3, y_3) = (3, y),$$

where y can take one of the four values $8, 3, -1, -8$. The least squares problem is, for every y, to determine the single parameter x, to minimize the function

$$f(x) = \tfrac{1}{2}\sum_{i=1}^{3} r_i(x)^2 = \tfrac{1}{2}\left[(e^x - 2)^2 + (e^{2x} - 4)^2 + (e^{3x} - y)^2\right].$$

For this function of a single variable x, both the simplified and full Hessian are scalars:

$$J(x)^T J(x) = \sum_{i=1}^{3} (t_i e^{xt_i})^2 \qquad \text{and} \qquad S(x) = \sum_{i=1}^{3} r_i(x) t_i^2 e^{xt_i}.$$

Table 9.1 lists, for each of the four values of y, the minimizer x^* and values of several functions at the minima.

We use the Newton and Gauss-Newton methods with several starting values x_0. Table 9.2 lists the different convergence behaviors for every case, with two different starting values. The stopping criterion for the iterations was $|\nabla f(x_k)| \le 10^{-10}$. Note that for the consistent problem with $y = 8$, Gauss-Newton achieves quadratic convergence, since $S(x^*) = 0$. For $y = 3$, the convergence factor for Gauss-Newton is $\sigma/\lambda \simeq 0.06$, which is small compared to 0.47 for $y = -1$, although for this value of y there is still linear but slow convergence. For $y = -8$ the ratio is $6.5 \gg 1$ and there is no convergence.

		Newton		Gauss-Newton	
y	x_0	# iter.	rate	# iter.	rate
8	1	7	quadratic	5	quadratic
	0.6	6	quadratic	4	quadratic
3	1	9	quadratic	12	linear
	0.5	5	quadratic	9	linear
−1	1	10	quadratic	34	linear (slow)
	0	4	quadratic	32	linear (slow)
−8	1	12	quadratic	no convergence	
	−0.7	4	quadratic	no convergence	

Table 9.2: Convergence behavior for the four cases and two different starting values.

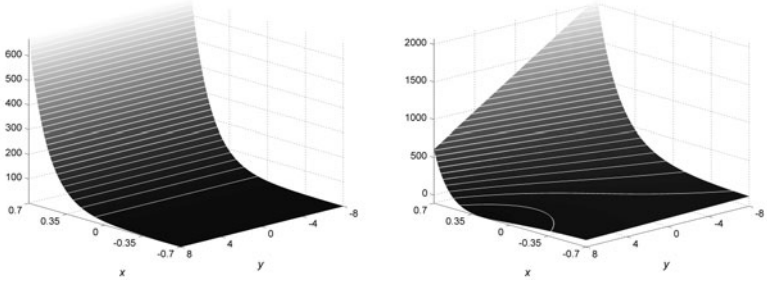

Figure 9.2.1: Single-parameter exponential fit example: the scalars $J(x)^T J(x)$ (left) and $\nabla^2 f(x)$ (right) as functions of x and y (the white lines are level curves). The second term in $\nabla^2 f(x)$ has increasing importance as y decreases.

To support the above results, Figure 9.2.1 shows plots of the Hessian $\nabla^2 f(x)$ and its first term $J(x)^T J(x)$ (recall that both are scalars) as functions of x and y, for $x \in [-0.7, 0.7]$ and $y \in [-8, 8]$. We see that the deterioration of the convergence rate of Gauss-Newton's method, compared to that of Newton's method, indeed coincides with the increasing importance of the term $S(x)$ in $\nabla^2 f(x)$, due to larger residuals at the solution as y decreases.

For large-scale problems, where memory and computing time are limiting factors, it may be infeasible to solve the linear LSQ problem for the correction $\Delta \boldsymbol{x}_k$ to high accuracy in each iteration. Instead one can use one of the iterative methods discussed in Chapter 6 to compute an approximate step $\triangle \tilde{\boldsymbol{x}}_k$, by terminating the iterations once the approximation is "accu-

rate enough," e.g., when the residual in the normal equations is sufficiently small:

$$\|J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)\,\Delta\tilde{\boldsymbol{x}}_k + J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k)\|_2 < \tau \|J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k)\|_2,$$

for some $\tau \in (0,1)$. In this case, the algorithm is referred to as an *inexact Gauss-Newton method*. For these nested iterations an approximate Newton method theory applies [61, 143, 183].

9.3 The Levenberg-Marquardt method

While the Gauss-Newton method can be used for ill-conditioned problems, it is not efficient, as one can see from the above convergence relations when $\lambda \to 0$. The main difficulty is that the correction $\Delta\boldsymbol{x}_k$ is too large and goes in a "bad" direction that gives little reduction in the function. For such problems, it is common to add an inequality constraint to the linear least squares problem (9.2.2) that determines the step, namely, that the length of the step $\|\Delta\boldsymbol{x}_k\|_2$ should be bounded by some constant. This so-called trust region technique improves the quality of the step.

As we saw in the previous chapter, we can handle such an inequality constraint via the use of a Lagrange multiplier and thus replace the problem in (9.2.2) with a problem of the form

$$\min_{\triangle \boldsymbol{x}_{k+1}} \left\{ \|J(\boldsymbol{x}_k)\,\Delta\boldsymbol{x}_k + \boldsymbol{r}(\boldsymbol{x}_k)\|_2^2 + \lambda_k \|\boldsymbol{x}_k\|_2^2 \right\},$$

where $\lambda_k > 0$ is the Lagrange multiplier for the constraint at the kth iteration. There are two equivalent forms of this problem, either the "modified" normal equations

$$\left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + \lambda_k I\right) \Delta\boldsymbol{x}_k = -J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k) \tag{9.3.1}$$

or the "modified" least squares problem

$$\min_{\Delta\boldsymbol{x}_k} \left\| \begin{pmatrix} J(\boldsymbol{x}_k) \\ \sqrt{\lambda_k} I \end{pmatrix} \Delta\boldsymbol{x}_k - \begin{pmatrix} -\boldsymbol{r}(\boldsymbol{x}_k) \\ \boldsymbol{0} \end{pmatrix} \right\|_2 . \tag{9.3.2}$$

The latter is best suited for numerical computations. This approach, which also handles a rank-deficient Jacobian $J(\boldsymbol{x}_k)$, leads to the Levenberg-Marquardt method, which takes the following form:

Algorithm Levenberg-Marquardt

- Start with an initial guess $\boldsymbol{x}_0$ and iterate for $k = 0, 1, 2, \ldots$
- At each step k choose the Lagrange multiplier λ_k.

- Solve 9.3.1 or 9.3.2 for $\Delta \boldsymbol{x}_k$.
- Calculate the next iterate $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta \boldsymbol{x}_k$.
- Check for convergence.

The parameter λ_k influences both the direction and the length of the step $\Delta \boldsymbol{x}_k$. Depending on the size of λ_k, the correction $\Delta \boldsymbol{x}_k$ can vary from a Gauss-Newton step for $\lambda_k = 0$, to a very short step approximately in the steepest descent direction for large values of λ_k. As we see from these considerations, the LM parameter acts similarly to the step control for the damped Gauss-Newton method, but it also changes the direction of the correction.

The Levenberg-Marquardt step can be interpreted as solving the normal equations used in the Gauss-Newton method, but "shifted" by a scaled identity matrix, so as to convert the problem from having an ill-conditioned (or positive semidefinite) matrix $J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)$ into a positive definite one. Notice that the positive definiteness implies that the Levenberg-Marquardt direction is always of descent and therefore the method is well defined.

Another way of looking at the Levenberg-Marquardt iteration is to consider the matrix $\sqrt{\lambda_k} I$ as an approximation to the second derivative term $S(\boldsymbol{x}_k)$ that was neglected in the definition of the Gauss-Newton method.

The local convergence of the Levenberg-Marquardt method is summarized in the following theorem.

Theorem 93. *Assume the same conditions as in Theorem 91 and in addition assume that the Lagrange multipliers λ_k for $k = 0, 1, 2, \ldots$ are nonnegative and bounded by $b > 0$. If $\sigma < \lambda$, then for any $c \in (1, (\lambda+b)/(\sigma+b))$, there exists an open ball D around $\boldsymbol{x}^*$ such that the Levenberg-Marquardt sequence converges linearly to $\boldsymbol{x}^*$, starting from any initial point $\boldsymbol{x}_0 \in \mathcal{D}$:*

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|_2 \leqslant \frac{c(\sigma + b)}{\lambda + b} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 + \frac{c\alpha\gamma}{2(\lambda + b)} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2^2$$

and

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|_2 \leqslant \frac{c(\sigma + b) + (\lambda + b)}{2(\lambda + b)} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 < \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 .$$

If $\boldsymbol{r}(\boldsymbol{x}^*) = \boldsymbol{0}$ and $\lambda_k = O\left(\left\|J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k)\right\|_2\right)$, then the iterates $\boldsymbol{x}_k$ converge quadratically to $\boldsymbol{x}^*$.

Within the trust-region framework introduced above, there are several general step-length determination techniques, see, for example, [170]. Here we give the original strategy devised by Marquardt for the choice of the parameter λ_k, which is simple and often works well. The underlying principles are

- The initial value λ_0 should be of the same size as $\|J(\boldsymbol{x}_0)^T J(\boldsymbol{x}_0)\|_2$.
- For subsequent steps, an improvement ratio is defined as in the trust region approach:

$$\varrho_k = \frac{\text{actual reduction}}{\text{predicted reduction}} = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})}{\frac{1}{2}\Delta\boldsymbol{x}_k^T(J(\boldsymbol{x}_k)^T\boldsymbol{r}(\boldsymbol{x}_k) - \lambda_k\Delta\mathbf{x}_k)}.$$

Here, the denominator is the reduction in f predicted by the local linear model. If ϱ_k is large, then the pure Gauss-Newton model is good enough, so λ_{k+1} can be made smaller than at the previous step. If ϱ_k is small (or even negative), then a short, steepest-descent step should be used, i.e., λ_{k+1} should be increased. Marquard's updating strategy is widely used with some variations in the thresholds.

Algorithm Levenberg-Marquardt's parameter updating

- If $\rho_k > 0.75$, then $\lambda_{k+1} = \lambda_k/3$.
- If $\rho_k < 0.25$, then $\lambda_k = 2\,\lambda_k$.
- Otherwise use $\lambda_{k+1} = \lambda_k$.
- If $\rho_k > 0$, then perform the update $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x}_k$.

A detailed description can be found in [154].

The software package MINPACK-1 available from Netlib [263] includes a robust implementation based on the Levenberg-Marquardt algorithm.

Similar to the Gauss-Newton method, we have inexact versions of the Levenberg-Marquardt method, where the modified least squares problem (9.3.2) for the correction is solved only to sufficient accuracy, i.e., for some $\tau \in (0, 1)$ we accept the solution if:

$$\|(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + \lambda_k I)\,\Delta\boldsymbol{x}_k + J(\boldsymbol{x}_k)^T\boldsymbol{r}(\boldsymbol{x}_k)\|_2 < \tau\|J(\boldsymbol{x}_k)^T\boldsymbol{r}(\boldsymbol{x}_k)\|_2.$$

If this system is to be solved for several values of the Lagrange parameter λ_k, then the bidiagonalization strategy from Section 5.5 can be utilized such that only one partial bidiagonalization needs to be performed; for more details see [121, 260].

Example 94. *We return to Example 92, and this time we use the Levenberg-Marquardt method with the above parameter-updating algorithm and the same starting values and stopping criterion as before. Table 9.3 compares the performance of Levenberg-Marquardt's algorithm with that of the Gauss-Newton algorithm. For the consistent problem with $y = 8$, the convergence is still quadratic, but slower. As the residual increases, the advantage of the Levenberg-Marquardt strategy sets in, and we are now able to solve the large-residual problem for $y = -8$, although the convergence is very slow.*

		Gauss-Newton		Levenberg-Marquardt	
y	x_0	# iter.	rate	# iter.	rate
8	1	5	quadratic	10	quadratic
	0.6	4	quadratic	7	quadratic
3	1	12	linear	13	linear
	0.5	9	linear	10	linear
-1	1	34	linear (slow)	26	linear (slow)
	0	32	linear (slow)	24	linear (slow)
-8	1	no convergence		125	linear (very slow)
	-0.7	no convergence		120	linear (very slow)

Table 9.3: Comparison of the convergence behavior of the Gauss-Newton and Levenberg-Marquardt methods for the same test problem as in Table 9.2.

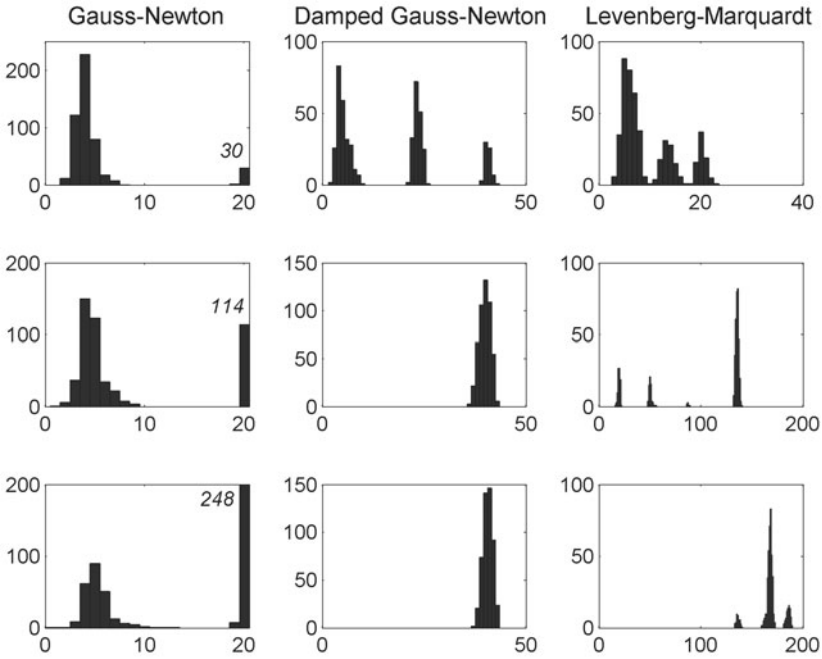

Figure 9.3.1: Number of iterations to reach an absolute accuracy of 10^{-6} in the solution by three NLLSQ different methods. Each histogram shows results for a particular method and a particular value of ζ in the perturbation of the starting point; top $\zeta = 0.1$, middle $\zeta = 0.2$, bottom $\zeta = 0.3$.

Example 95. *This example is mainly concerned with a comparison of the robustness of the iterative methods with regards to the initial guess* $\boldsymbol{x}_0$*. We use the Gaussian model from Examples 83 and 86 with noise level* $\varsigma = 0.1$*, and we use three different algorithms to solve the NLLSQ problem:*

- *The standard Gauss-Newton method.*
- *An implementation of the damped Gauss-Newton algorithm from MATLAB's Optimization Toolbox, available via the options* `'LargeScale'='off'` *and* `'LevenbergMarquardt'='off'` *in the function* `lsqnonlin`*.*
- *An implementation of the Levenberg-Marquardt algorithm from MATLAB's Optimization Toolbox, available via the options* `'Jacobian'='on'` *and* `'Algorithm'='levenberg-marquardt'` *in the function* `lsqnonlin`*.*

The starting guess was chosen equal to the exact parameters $(2.2,\ 0.26,\ 0.2)^T$ *plus a Gaussian perturbation with standard deviation* ζ*. We used* $\zeta = 0.1$*,* 0.2*,* 0.3*, and for each value we created 500 different realizations of the initial point. Figure 9.3.1 shows the number of iterations in the form of histograms – notice the different axes in the nine plots. We give the number of iterations necessary to reach an absolute accuracy of* 10^{-6}*, compared to a reference solution computed with much higher accuracy. The italic numbers in the three left plots are the number of times the Gauss-Newton method did not converge.*

We see that when the standard Gauss-Newton method converges, it converges rapidly – but also that it may not converge. Moreover, as the starting point moves farther from the minimizer, the number of instances of non-convergence increases dramatically.

The damped Gauss-Newton method used here always converged, thus was much more robust than the undamped version. For starting points close to the minimum, it may converge quickly, but it may also require about 40 iterations. For starting points further away from the solution, this method always uses about 40 iterations.

The Levenberg-Marquardt method used here is also robust, in that it converges for all starting points. In fact, for starting points close to the solution it requires, on the average, fewer iterations than the damped Gauss-Newton method. For starting points farther from the solution it still converges, but requires many more iterations. The main advantage of the Levenberg-Marquardt algorithm, namely, to handle ill-conditioned and rank-deficient Jacobian matrices, does not come into play here as the particular problem is well conditioned.

We emphasize that this example should not be considered as representative for these methods in general – rather, it is an example of the perfor-

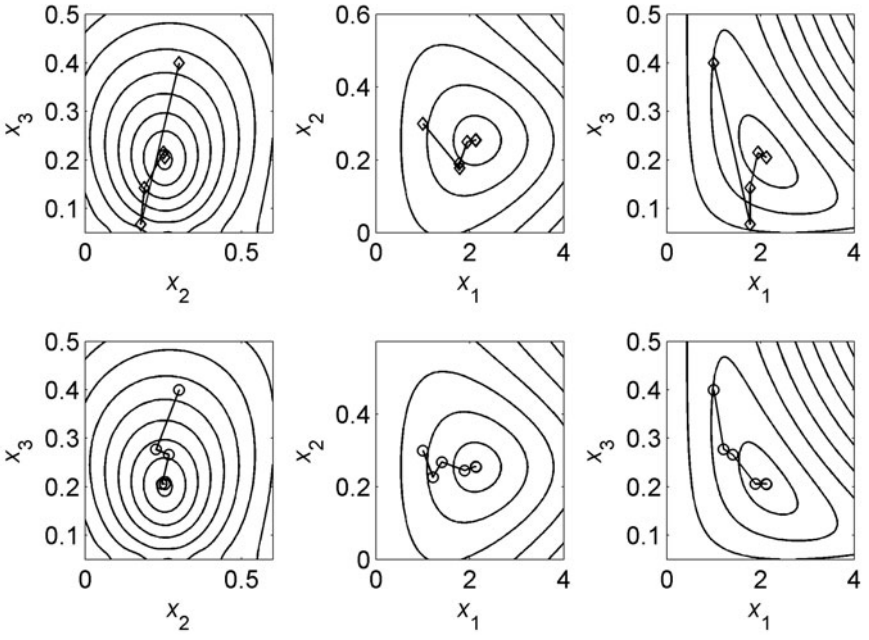

Figure 9.3.2: Convergence histories from the solution of a particular NLLSQ problem with three parameters by means of the standard Gauss-Newton (top) and the Levenberg-Marquardt algorithm (bottom). Each figure shows the behavior of pairs of the three components of the iterates $\boldsymbol{x}_k$ together with level sets of the objective function.

mance of specific implementations for one particular (and well-conditioned) problem.

Example 96. *We illustrate the robustness of the Levenberg-Marquardt algorithm by solving the same test problem as in the previous example by means of the standard Gauss-Newton and Levenberg-Marquardt algorithms. The progress of a typical convergence history for these two methods is shown in Figure 9.3.2. There are three unknowns in the model, and the starting point is* $\boldsymbol{x}_0 = (1, 0.3, 0.4)^T$. *The top figures show different component pairs of the iterates* $\boldsymbol{x}_k$ *for* $k = 1, 2, \ldots$ *for the Gauss-Newton algorithm (for example, the leftmost plot shows the third component of* $\boldsymbol{x}_k$ *versus its second component). Clearly, the Gauss-Newton iterates overshoot before they finally converge to the LSQ solution. The iterates of the Levenberg-Marquardt algorithm are shown in the bottom figures, and we see that they converge much faster toward the LSQ solution without any big "detour."*

Characteristics	Newton	G-N	L-M
Ill-conditioned Jacobian	yes	yes (but slow)	yes
Rank-deficient Jacobian	yes	no	yes
Convergence $S(\boldsymbol{x}_k) = 0$	quadratic	quadratic	quadratic
Convergence $S(\boldsymbol{x}_k)$ small	quadratic	linear	linear
Convergence $S(\boldsymbol{x}_k)$ large	quadratic	slow or none	slow or none

Table 9.4: Comparison of some properties of the Newton, Gauss-Newton (G-N) and Levenberg-Marquardt (L-M) algorithms for nonlinear least squares problems.

9.4 Additional considerations and software

An overall comparison between the methods discussed so far is given in Table 9.4. The "yes" or "no" indicates whether the corresponding algorithm is appropriate or not for the particular problem. Below we comment on some further issues that are relevant for these methods.

Hybrid methods

During the iterations we do not know whether we are in the region where the convergence conditions of a particular method hold, so the idea in hybrid methods is to combine a "fast" method, such as Newton (if second derivatives are available), with a "safe" method such as steepest descent. One hybrid strategy is to combine Gauss-Newton or Levenberg-Marquardt, which in general is only linearly convergent, with a superlinearly convergent quasi-Newton method, where $S(\boldsymbol{x}_k)$ is approximated by a secant term.

An example of this efficient hybrid method is the algorithm NL2SOL by Dennis, Gay and Welsch [65]. It contains a trust-region strategy for global convergence and uses Gauss-Newton and Levenberg-Marquardt steps initially, until it has enough good second-order information. Its performance for large and very nonlinear problems is somewhat better than Levenberg-Marquardt, in that it requires fewer iterations. For more details see [66].

Starting and stopping

In many cases there will be no good a priori estimates available for the initial point x_0. As mentioned in the previous section, nonlinear least squares problems are usually non-convex and may have several local minima. Therefore some global optimization technique must be used to descend from undesired local minima if the global minimum is required. One possibility is a simple Monte Carlo strategy, in which multiple initial points are chosen at random and the least squares algorithm is started several times. It is

hoped that some of these iterations will converge, and one can then choose the best solution. See, e.g., [131] for an overview of global optimization algorithms.

In order to make such a process more efficient, especially when function evaluations are costly, a procedure has been described in [191] that saves all iterates and confidence radiuses. An iteration is stopped if an iterate lands in a previously calculated iterate's confidence sphere. The assumption underlying this decision is that the iteration will lead to a minimum already calculated. The algorithm is trivially parallelizable and runs very efficiently in a distributed network. Up to 40% savings have been observed from using the early termination strategy. This simple approach is easily parallelizable, but it can only be applied to low-dimensional problems.

Several criteria ought to be taken into consideration for stopping the iterative methods described above.

- The sequence convergence criterion: $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2 \leq$ tolerance (not a particularly good one)!
- The consistency criterion: $|f(\boldsymbol{x}_{k+1})| \leq$ tolerance (relevant only for problems with $\boldsymbol{r}(\boldsymbol{x}^*) = \boldsymbol{0}$).
- The absolute function criterion: $\|\nabla f(\boldsymbol{x}_{k+1})\|_2 \leq$ tolerance.
- The maximum iteration count criterion: $k > k_{\max}$.

The absolute function criterion has a special interpretation for NLLSQ problems, namely, that the residual at $\boldsymbol{x}_{k+1}$ is nearly orthogonal to the subspace generated by the columns of the Jacobian. For the Gauss-Newton and Levenberg-Marquardt algorithms, the necessary information to check near-orthogonality is easily available.

Methods for step-length determination

Control of the step length is important for ensuring a robust algorithm that converges from starting points far from the minimum. Given an iterate $\boldsymbol{x}_k$ and a descent direction $\boldsymbol{p}$, there are two common ways to choose the step-length α_k.

- Take α_k as the solution to the one-dimensional minimization problem

$$\min_\alpha \|\boldsymbol{r}(\boldsymbol{x}_k + \alpha_k \boldsymbol{p})\|_2.$$

  This is expensive if one tries to find the exact solution. Fortunately, this is not necessary, and so-called soft or inexact line search strategies can be used.

- Inexact line searches: an α_k is accepted if a sufficient descent condition is satisfied for the new point $\boldsymbol{x}_k + \alpha_k \boldsymbol{p}$ and if α_k is large enough that there is a gain in the step. Use the so-called Armijo-Goldstein step-length principle, where α_k is chosen as the largest number from the sequence $\alpha = 1, \frac{1}{2}, \frac{1}{4}, \ldots$ for which the inequality

$$\|\boldsymbol{r}(\boldsymbol{x}_k)\|_2 - \|\boldsymbol{r}(\boldsymbol{x}_k + \alpha_k \boldsymbol{p})\|_2 \geq \tfrac{1}{2}\alpha_k \|J(\boldsymbol{x}_k)\,\boldsymbol{p}\|_2$$

  holds.

For details see, for example, chapter 3 in [170] and for a brief overview see [20] p. 344.

Software

In addition to the software already mentioned, the PORT library, available from AT&T Bell Labs and partly from Netlib [263], has a range of codes for nonlinear least squares problems, some requiring the Jacobian and others that need only information about the objective function. The Gauss-Newton algorithm is not robust enough to be used on its own, but most of the software for nonlinear least squares that can be found in NAG, MATLAB and TENSOLVE (using tensor methods), have enhanced Gauss-Newton codes. MINPACK-1, MATLAB and IMSL contain Levenberg-Marquardt algorithms. The NEOS Guide on the Web [264] is a source of information about optimization in general, with a section on NLLSQ. Also, the National Institute of Standards and Technology Web page at [265] is very useful, as well as the book [165].

Some software packages for large-scale problems with a sparse Jacobian are VE10 and LANCELOT. VE10 [264], developed by P. Toint, implements a line search method, where the search direction is obtained by a truncated conjugate gradient technique. It uses an inexact Newton method for partially separable nonlinear problems with sparse structure. LANCELOT [148] is a software package for nonlinear optimization developed by A. Conn, N. Gould and P. Toint. The algorithm combines a trust-region approach, adapted to handle possible bound constraints and projected gradient techniques. In addition it has preconditioning and scaling provisions.

9.5 Iteratively reweighted LSQ algorithms for robust data fitting problems

In Chapter 1 we introduced the robust data fitting problem based on the principle of M-estimation, leading to the nonlinear minimization problem $\min_{\boldsymbol{x}} \sum_{i=1}^{m} \rho\left(r_i(\boldsymbol{x})\right)$ where the function ρ is chosen so that it gives less

weight to residuals $r_i(\boldsymbol{x})$ with large absolute value. Here we consider the important case of robust linear data fitting, where the residuals are the elements of the residual vector $\boldsymbol{r} = \boldsymbol{b} - A\,\boldsymbol{x}$. Below we describe several iterative algorithms that, in spite of their differences, are commonly referred to as *iteratively reweighted least squares algorithms* due to their use of a sequence of linear least squares problems with weights that change during the iterations.

To derive the algorithms we consider the problem of minimizing the function

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \rho\left(r_i(\boldsymbol{x})\right), \qquad \boldsymbol{r} = \boldsymbol{b} - A\,\boldsymbol{x}.$$

We introduce the vector $\boldsymbol{g} \in \mathbb{R}^m$ and the diagonal matrix $D \in \mathbb{R}^{m\times m}$ with elements defined by

$$g_i = \rho'(r_i), \qquad d_{ii} = \rho''(r_i), \qquad i = 1, \ldots, m, \tag{9.5.1}$$

where ρ' and ρ'' denote the first and second derivatives of $\rho(r)$ with respect to r. Then the gradient and the Hessian of f are given by

$$\nabla f(\boldsymbol{x}) = -A^T \boldsymbol{g}(\boldsymbol{x}) \qquad \text{and} \qquad \nabla^2 f(\boldsymbol{x}) = A^T D(\boldsymbol{x})\, A, \tag{9.5.2}$$

where we use the notation $\boldsymbol{g}(\boldsymbol{x})$ and $D(\boldsymbol{x})$ to emphasize that these quantities depend on $\boldsymbol{x}$.

The original iteratively reweighted least squares algorithm [8] uses a fixed-point iteration to solve $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$. From (9.5.1) and (9.5.2) and introducing the diagonal $m \times m$ weight matrix

$$W(\boldsymbol{r}) = \operatorname{diag}\left(\rho'(r_i)/r_i\right) \qquad \text{for} \qquad i = 1, \ldots, m, \tag{9.5.3}$$

we obtain the nonlinear equations

$$A^T \boldsymbol{g}(\boldsymbol{x}) = A^T W(\boldsymbol{r})\, \boldsymbol{r} = A^T W(\boldsymbol{r})\, (\boldsymbol{b} - A\,\boldsymbol{x}) = \boldsymbol{0}$$

or

$$A^T W(\boldsymbol{r})\, A\,\boldsymbol{x} = A^T W(\boldsymbol{r})\, \boldsymbol{b}.$$

The fixed-point iteration scheme used in [8] is

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \left(A^T W(\boldsymbol{r}_k)\, A\right)^{-1} A^T W(\boldsymbol{r}_k)\, \boldsymbol{b} \\ &= \operatorname{argmin}_{\boldsymbol{x}} \| W(\boldsymbol{r}_k)^{1/2}\, (\boldsymbol{b} - A\,\boldsymbol{x}_k) \|_2, \quad k = 1, 2, \ldots \end{aligned} \tag{9.5.4}$$

where $\boldsymbol{r}_k = \boldsymbol{b} - A\,\boldsymbol{x}_k$ is the residual from the previous iteration. Hence, each new iterate is the solution of a weighted least squares problem, in which the weights depend on the solution from the previous iteration. This algorithm, with seven different choices of the function ρ, is implemented

in a Fortran software package described in [50] where more details can be found.

A faster algorithm for solving the robust linear data fitting problem – also commonly referred to as an iteratively reweighted least squares algorithm – is obtained by applying Newton's method from Section 9.1 to the function $f(\boldsymbol{x}) = \sum_{i=1}^{m} \rho\left(r_i(\boldsymbol{x})\right)$; see ([20], section 4.5.3) for details. According to (9.5.2), the Newton iteration is

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left(A^T D(\boldsymbol{x}_k)\, A\right)^{-1} A^T \boldsymbol{g}(\boldsymbol{x}_k), \qquad k = 1, 2, \ldots . \tag{9.5.5}$$

We emphasize that the Newton update is not a least squares solution, because the diagonal matrix $D(\boldsymbol{x}_k)$ appears in front of the vector $\boldsymbol{g}(\boldsymbol{x}_k)$. O'Leary [171] suggests a variant of this algorithm where, instead of updating the solution vectors, the residual vector is updated as

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k\, A \left(A^T D(\boldsymbol{x}_k)\, A\right)^{-1} A^T \boldsymbol{g}(\boldsymbol{x}_k), \qquad k = 1, 2, \ldots, \tag{9.5.6}$$

and the step-length α_k is determined via line search. Upon convergence the robust solution $\boldsymbol{x}_k$ is computed from the final residual vector $\boldsymbol{r}_k$ as the solution to the consistent system $A\,\boldsymbol{x} = \boldsymbol{b} - \boldsymbol{r}_k$. Five different numerical methods for computing the search direction in (9.5.6) are compared in [171], where it is demonstrated that the choice of the best method depends on the function ρ. Wolke and Schwetlick [258] extend the algorithm to also estimate the parameter β that appears in the function ρ and which must reflect the noise level in the data.

As starting vector for the above iterative methods one can use the ordinary LSQ solution or, alternatively, the solution to the 1-norm problem $\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_1$ (see below).

The iteratively reweighted least squares formalism can also be used to solve linear p-norm problem $\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_p$ for $1 < p < 2$. To see this, we note that

$$\begin{aligned} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_p^p &= \sum_{i=1}^{m} |r_i(\boldsymbol{x})|^p = \sum_{i=1}^{m} |r_i(\boldsymbol{x})|^{p-2} r_i(\boldsymbol{x})^2 \\ &= \|W_p(\boldsymbol{r})^{1/2}\,(\boldsymbol{b} - A\,\boldsymbol{x})\|_2^2 \end{aligned}$$

where $W_p(\boldsymbol{r}) = \mathrm{diag}(|r_i|^{p-2})$. The iterations of the corresponding Newton-type algorithm take the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x}_k, \qquad k = 1, 2, \ldots , \tag{9.5.7}$$

where $\Delta\boldsymbol{x}_k$ is the solution to the weighted LSQ problem

$$\min_{\Delta\boldsymbol{x}} \|W_p(\boldsymbol{r}_k)^{1/2}\,(\boldsymbol{r}_k - A\,\Delta\boldsymbol{x})\|_2, \tag{9.5.8}$$

see (Björck [20], section 4.5.2) for details. Experience shows that, for p close to 1, the LSQ problem (9.5.8) tends to become ill-conditioned as the iterations converge to the robust solution. Special algorithms are needed for the case $p = 1$; some references are [6, 51, 153].

For early use in geophysical prospecting see [48, 219].

9.6 Variable projection algorithm

We finish this chapter with a brief discussion of algorithms to solve the separable NLLSQ problem introduced in Section 8.4, where we listed some important properties of this problem. The simplest method of solution that takes into account the special structure is the algorithm NIPALS [256], where an alternating optimization procedure is employed: the linear and nonlinear parameters are successively fixed and the problem is minimized over the complementary set. This iteration, however, only converges linearly.

The *variable projection* algorithm, developed by Golub and Pereyra [101], takes advantage instead of the explicit coupling between the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{a}$, in order to reduce the original minimization problem to two subproblems that are solved in sequence, without alternation. The nonlinear subproblem is smaller (although generally more complex) and involves only the k parameters in $\boldsymbol{\alpha}$. One linear subproblem is solved a posteriori to determine the n linear parameters in $\boldsymbol{a}$. The difference with NIPALS is perhaps subtle, but it makes the algorithm much more powerful, as we will see below.

For the special case when $\phi_j(\boldsymbol{\alpha}, t) = e^{\alpha_j t}$, the problem is called *exponential data fitting,* which is a very common and important application that is notoriously difficult to solve. Fast methods that originated with Prony [204] are occasionally used, but, unfortunately, the original method is not suitable for problems with noise. The best-modified versions are from M. Osborne [175, 176, 177], who uses the separability to achieve better results. As shown in [198], the other method frequently used with success is the variable projection algorithm. A detailed discussion of both methods and their relative merits is found in chapter 1 of [198], while the remaining chapters show a number of applications in very different fields and where either or both methods are used and compared.

The key idea behind the variable projection algorithm is to eliminate the linear parameters analytically by using the pseudoinverse, solve for the nonlinear parameters first and then solve a linear LSQ problem for the remaining parameters. Figure 9.6.1 illustrates the variable projection principle in action. We depict $I - \Phi(\boldsymbol{\alpha})\Phi^{\dagger}(\boldsymbol{\alpha})$ for a fixed $\boldsymbol{\alpha}$ as a linear mapping from $\mathbb{R}^n \to \mathbb{R}^m$. That is, its range is a linear subspace of dimension n (or less, if the matrix is rank deficient). When $\boldsymbol{\alpha}$ varies, this subspace

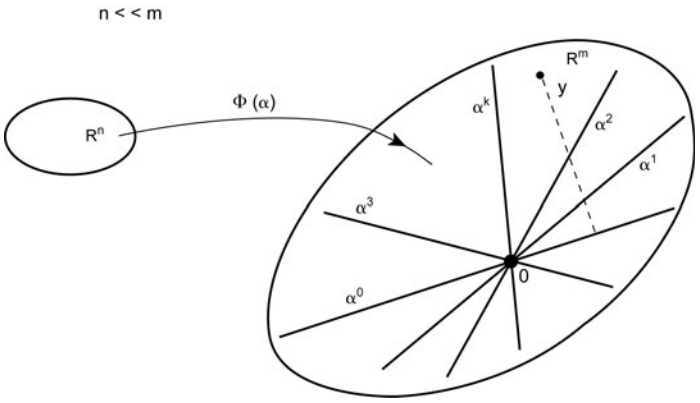

Figure 9.6.1: The geometry behind the variable projection principle. For each $\boldsymbol{\alpha}^i$ the projection $I - \Phi(\boldsymbol{\alpha})\Phi^\dagger(\boldsymbol{\alpha})$ maps $\mathbb{R}^n$ into a subspace in $\mathbb{R}^m$ (depicted as a line).

pivots around the origin. For each $\boldsymbol{\alpha}$ the residual is equal to the Euclidean distance from the data vector $\boldsymbol{y}$ to the corresponding subspace. As usual, there may not be any subspace to which the data belong, i.e., the problem is inconsistent and there is a nonzero residual at the solution. This residual is related to the l_2 approximation ability of the basis functions $\phi_j(\boldsymbol{\alpha}, t)$.

There are several important results proved in [101], [227] and [212], which show that the reduced function is better conditioned than the original one. Also, we observe that

- The reduction in dimensionality of the problem has as a consequence that fewer initial parameters need to be guessed to start a minimization procedure.

- The algorithm is valid in the rank-deficient case. To guarantee continuity of the Moore-Penrose generalized inverse, only local constant rank of $\Phi(\boldsymbol{\alpha})$ needs to be assumed.

- The reduced nonlinear function, although more complex and therefore apparently costlier to evaluate, gives rise to a better-conditioned problem, which always takes fewer iterations to converge than the full problem [212]. This may include convergence when the Levenberg-Marquardt algorithm for the full function does not converge.

- By careful implementation of the linear algebra involved and use of a simplification due to Kaufman [140], the cost per iteration for the reduced function is similar to that for the full function, and thus minimization of the reduced problem is always faster. However, in

hard problems this simplification may lead to a less robust algorithm [172].

The linear problem is easily solved by using the methods discussed in previous chapters. There are two types of iterative methods to solve the NLLSQ problem in the variable projection algorithm:

- **Derivative free methods** such as PRAXIS [263] require only an efficient computation of the nonlinear function

$$f_2(\boldsymbol{\alpha}) = \tfrac{1}{2}\|\boldsymbol{y} - \Phi(\boldsymbol{\alpha})\Phi(\boldsymbol{\alpha})^{\dagger}\,\boldsymbol{y}\|_2^2 = \tfrac{1}{2}\|P_{\Phi(\boldsymbol{\alpha})}\,\boldsymbol{y}\|_2^2.$$

  Instead of the more expensive pseudoinverse computation it is possible to obtain $P_{\Phi(\boldsymbol{\alpha})}$ by orthogonally transforming $\Phi(\boldsymbol{\alpha})$ into trapezoidal form. One obtains then a simple expression for the evaluation of the function from the same orthogonal transformation when applied to $\boldsymbol{y}$. For details see [101].

- **Methods that need derivatives** of $\boldsymbol{r}(\boldsymbol{\alpha}) = \boldsymbol{y} - \Phi(\boldsymbol{\alpha})\Phi(\boldsymbol{\alpha})^{\dagger}\,\boldsymbol{y}$. In [101], a formula for the Fréchet derivative of the orthogonal projector $P_{\Phi(\boldsymbol{\alpha})}$ was developed and then used in the Levenberg-Marquardt algorithm, namely:

$$D\boldsymbol{r}(\boldsymbol{\alpha}) = -[P^{\perp}_{\Phi(\boldsymbol{\alpha})}D(\Phi(\boldsymbol{\alpha}))\Phi(\boldsymbol{\alpha})^{\dagger} + (P^{\perp}_{\Phi(\boldsymbol{\alpha})}D(\Phi(\boldsymbol{\alpha}))\Phi(\boldsymbol{\alpha})^{\dagger})^{T}]\boldsymbol{y}.$$

  Kaufman [140] ignores the second term, producing a saving of up to 25% in computer time, without a significant increase in the number of iterations. The Levenberg-Marquardt algorithm used in [101] and [140] starts with an arbitrary $\boldsymbol{\alpha}_0$ and determines the iterates from the relation:

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k - \begin{pmatrix} D\boldsymbol{r}(\boldsymbol{\alpha}_k) \\ \sqrt{\lambda_k}I \end{pmatrix}^{\dagger} \begin{pmatrix} \boldsymbol{r}(\boldsymbol{\alpha}_k) \\ \boldsymbol{0} \end{pmatrix}.$$

  At each iteration, this linear LSQ problem is solved by orthogonal transformations. Also, the Marquardt parameter λ_k can be determined so that divergence is prevented by enforcing descent:

$$\|\boldsymbol{r}(\boldsymbol{\alpha}_{k+1})\|_2 < \|\boldsymbol{r}(\boldsymbol{\alpha}_k)\|_2.$$

The original VARPRO program by Pereyra is listed in a Stanford University report [100]; there the minimization is done via the Osborne modification of the Levenberg-Marquardt algorithm. A public domain version, including modifications and additions by John Bolstadt, Linda Kaufman and Randy LeVeque, can be found in Netlib [263] under the same name. It incorporates the Kaufman modification, where the second term in the Fréchet derivative is ignored and the information provided by the program is used to generate a statistical analysis, including uncertainty bounds for

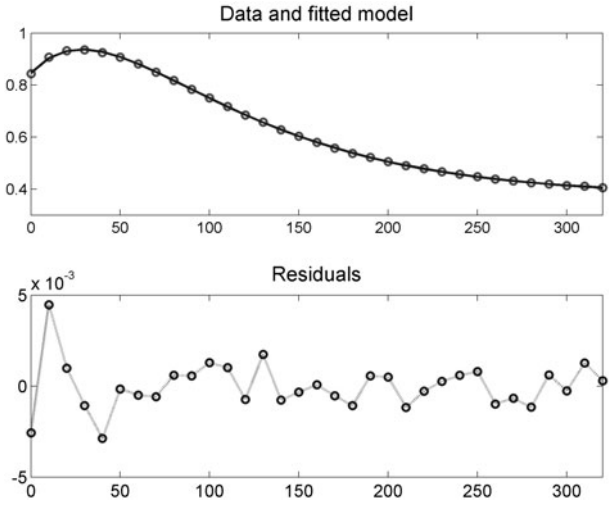

Figure 9.6.2: Data, fit and residuals for the exponential fitting problem.

the estimated linear and nonlinear parameters. Recent work by O'Leary and Rust indicates that in certain problems the Kaufman modification can make the algorithm less robust. They present in that work a modern and more modular implementation. In the PORT library [86] of Netlib there are careful implementations by Gay and Kaufman of variable projection versions for the case of unconstrained and constrained separable NLLSQ problems, including the option of using finite differences to approximate the derivatives. VARP2 [98, 141] is an extension for problems with multiple right-hand sides and it is also available in Netlib.

The VARPRO program was influenced by the state of the art in computing at the time of its writing, leading to a somewhat convoluted algorithm, and in a way most of the sequels inherited this approach. The computational constraints in memory and operation speeds have now been removed, since for most problems to which the algorithm is applicable in standard current machinery, results are produced quickly, even if multiple runs are executed in order to try to get a global minimum. In [198, 228] there is a description of a simplified approach for the case of complex exponentials, where some efficiency is sacrificed in order to achieve a clearer implementation that is easier to use and maintain.

Example 97. *The following example from [174] illustrates the competitiveness of the variable projection method, both in speed and robustness. Given a set of* $m = 33$ *data points* (t_i, y_i) *(the data set is listed in the VARPRO*

code in Netlib), fit the exponential model:

$$M(\boldsymbol{a}, \boldsymbol{\alpha}, t) = a_1 + a_2^{\alpha_1 t} + a_3^{\alpha_2 t}.$$

We compare the Levenberg-Marquardt algorithm for minimization of the full function with VARPRO, which uses the reduced function. The two algorithms needed 32 and 8 iterations, respectively, to reduce the objective function to less than $5 \cdot 10^{-5}$; this is a substantial saving, considering that the cost per iteration is similar for the two approaches. Moreover, the condition numbers of the respective linearized problems close to the solution are 48845 and 6.9; a clear example showing that the reduced problem is better conditioned than the original one. Figure 9.6.2 shows the fit and the residuals for the VP algorithm. The autocorrelation analysis from Section 1.4 gives $\varrho = -5 \cdot 10^{-6}$ and $T_\varrho = 9.7 \cdot 10^{-6}$, showing that the residuals can be considered as uncorrelated and therefore that the fit is acceptable. The least squares solution for this problem is

$$\boldsymbol{a}^* = (0.375,\ -1.46,\ 1.94)^T, \quad \boldsymbol{\alpha}^* = (0.0221,\ 0.0129)^T.$$

As explained in Section 3.5, the sensitivity of this solution can be assessed via the diagonal elements of the estimated covariance matrix

$$\varsigma^2 \big(J(\boldsymbol{x}^*)^T J(\boldsymbol{x}^*)\big)^{-1}.$$

In particular, the square roots of the diagonal elements of this matrix are estimates of the standard deviations of the five parameters. If we use the estimate $\varsigma^2 \simeq \|\boldsymbol{r}(\boldsymbol{x}^*)\|_2^2/m$, then these estimated standard deviations are

$$0.00191, \quad 0.204, \quad 0.203, \quad 0.000824, \quad 0.000413,$$

showing that the two linear parameters a_2 and a_3 are potentially very sensitive to perturbations. To illustrate this, the two alternative sets of parameters

$$\hat{\boldsymbol{a}} = (0.374,\ -1.29,\ 1.76)^T, \quad \hat{\boldsymbol{\alpha}} = (0.0231,\ 0.0125)^T$$

and

$$\tilde{\boldsymbol{a}} = (0.378,\ -2.29,\ 2.76)^T, \quad \tilde{\boldsymbol{\alpha}} = (0.0120,\ 0.0140)^T,$$

both give fits that are almost indistinguishable from the least squares fit. The corresponding residual norms are

$$\|\boldsymbol{r}(\boldsymbol{x}^*)\|_2 = 7.39 \cdot 10^{-3}, \quad \|\boldsymbol{r}(\hat{\boldsymbol{x}})\|_2 = 7.79 \cdot 10^{-3}, \quad \|\boldsymbol{r}(\tilde{\boldsymbol{x}})\|_2 = 8.24 \cdot 10^{-3},$$

showing that the large changes in a_2 and a_3 give rise to only small variations in the residuals, again demonstrating the lack of sensitivity of the residual to changes in those parameters.

9.7 Block methods for large-scale problems

We have already mentioned the inexact Gauss-Newton and Levenberg-Marquardt methods, based on truncated iterative solvers, as a way to deal with large computational problems. A different approach is to use a divide-and-conquer strategy that, by decomposing the problem into blocks, may allow the use of standard solvers for the (smaller) blocks. First, one subdivides the observations in appropriate non-overlapping groups. Through an SVD analysis one can select those variables that are more relevant to each subset of data; details of one such method for large-scale, ill-conditioned nonlinear least squares problems are given below and in [188, 189, 191].

The procedure works best if the data can be broken up in such a way that the associated variables have minimum overlap and only weak couplings are left with the variables outside the block. Of course, we cannot expect the blocks to be totally uncoupled; otherwise, the problem would decompose into a collection of problems that can be independently solved.

Thus, in general, the procedure consists of an outer block nonlinear Gauss-Seidel or Jacobi iteration, in which the NLLSQ problems corresponding to the individual blocks are solved approximately for their associated parameters. The block solver is initialized with the current value of the variables. The full parameter set is updated either after each block is processed (Gauss-Seidel strategy of information updating as soon as it is available), or after all the block solves have been completed (Jacobi strategy). The ill-conditioning is robustly addressed by the use of the Levenberg-Marquardt algorithm for the subproblems and by the threshold used to select the variables for each block.

The pre-processing for converting a large problem into block form starts by scanning the data and subdividing it. The ideal situation is one in which the data subsets are only sensitive to a small subset of the variables. Having performed that subdivision, we proceed to analyze the data blocks to determine which parameters are actually well determined by each subset of data. During this data analysis phase we compute the SVD of the Jacobian of each block; this potentially very large matrix is trimmed by deleting columns that are zero or smaller than a given threshold. Finally, the right singular vectors of the SVD are used to complete the analysis.

Selecting subspaces through the SVD

Jupp and Vozoff [138] introduced the idea of relevant and irrelevant parameters based on the SVD. We write first the Taylor expansion of the residual vector at a given point $\boldsymbol{x}$ (see §8.2):

$$\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h}) = \boldsymbol{r}(\boldsymbol{x}) + J(\boldsymbol{x})\,\boldsymbol{h} + O(\|\boldsymbol{h}\|_2^2). \tag{9.7.1}$$

Considering the SVD of the Jacobian $J(\boldsymbol{x}) = U\,\Sigma\,V^T$, we can introduce the so-called rotated perturbations

$$\boldsymbol{p} = \sigma_1 V^T \boldsymbol{h},$$

where σ_1 is the largest singular value of $J(\boldsymbol{x})$. Neglecting higher-order terms in (9.7.1) we can write this system as

$$\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h}) - \boldsymbol{r}(\boldsymbol{x}) = J(\boldsymbol{x})\,\boldsymbol{h} = \sum_{i=1}^{r} \left(\frac{\sigma_i}{\sigma_1}\right) p_i\,\boldsymbol{u}_i,$$

where σ_i/σ_1 are the normalized singular values and r is the rank of the Jacobian. This shows the direct relationship between the normalized singular values, the rotated perturbations $p_i = \sigma_1 \boldsymbol{v}_i^T \boldsymbol{h}$, and their influence on the variation of the residual vector. Thus,

$$\|\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h}) - \boldsymbol{r}(\boldsymbol{x})\|_2^2 = \sum_{i=1}^{r} \left(\frac{\sigma_i}{\sigma_1}\right)^2 p_i^2,$$

which shows that those parameters p_i that are associated with small normalized singular values will not contribute much to variations in the residual. Here we have assumed that all the components of the perturbation vector $\boldsymbol{h}$ are of similar size.

The above analysis is the key to the algorithm for partitioning the parameter set into blocks, once a partition of the data set has been chosen. Let RI_k denote the row indices for the kth block of data with $m^{[k]}$ elements. Given a base point $\boldsymbol{x}$, calculate the Jacobian $J(\boldsymbol{x})^{[k]}$ for this data set, i.e., $J(\boldsymbol{x})^{[k]}$ has only $m^{[k]}$ rows and less than n columns, since if the data have been partitioned appropriately and due to the local representation of the model, we expect that there will be a significant number of columns with small components that can be safely neglected. Then the procedure is as follows:

1. Compute the SVD of $J(\boldsymbol{x})^{[k]}$ and normalize the singular values by dividing them by the largest one.

2. Select the first $n^{[k]}$ normalized singular values that are above a given threshold, and their associated right singular vectors.

3. Inspect the set of chosen singular vectors and select the largest components of $V^{[k]}$in absolute value.

4. Choose the indices of the variables in parameter space corresponding to the columns of $V^{[k]}$ that contain large entries to form the set CI_k of column indices. This selects the subset of parameters that have most influence on the variation in the misfit functional for the given data set.

With this blocking strategy, variables may be repeated in different subsets. Observe also that it is possible for the union of all the subsets to be smaller than the entire set of variables; this will indicate that there are variables that cannot be resolved by the given data, at least in a neighborhood of the base point $\boldsymbol{x}$ and for the chosen threshold. Since this analysis is local, it should be periodically revised as the optimization process advances.

Once this partition has been completed, we use an outer block nonlinear Gauss-Seidel or Jacobi iteration [173] in order to obtain the solution of the full problem. To make this more precise, let us partition the index sets $M = \{1, 2, \ldots, m\}$ and $N = \{1, 2, \ldots, n\}$ into the subsets $\{\mathsf{RI}_k\}$ and $\{\mathsf{CI}_k\}$, i.e., the index sets that describe the partition of our problem into blocks. Each subproblem can now be written as

$$\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})^{[k]}\|_2^2 \;\text{ subject to }\; x = x_i^*, \quad i \in \{1, 2, \ldots, n\} - \mathsf{CI}_k, \tag{9.7.2}$$

where $\boldsymbol{r}(\boldsymbol{x})^{[k]}$ is the sub-vector of $\boldsymbol{r}(\boldsymbol{x})$ with elements $\{r(\boldsymbol{x})_i\}_{i \in \mathsf{RI}_k}$. In other words, we fix the values of the parameters that are *not* in block k to their current values in the global set of variables x^*. Observe that the dimension of the subproblem is then $m^{[k]} \times n^{[k]}$. By considering enough subsets $k = 1, \ldots, K$ we can make these dimensions small, especially $n^{[k]}$ and therefore make the subproblems (9.7.2) amenable to direct techniques (and global optimization, if necessary).

One step of a sequential block Gauss-Seidel iteration consists then in sweeping over all the blocks, solving the subproblems to a certain level of accuracy, and replacing the optimal values in the central repository of all the variables at once. A sequential block Jacobi iteration does the same, but it does not replace the values until the sweep over all the blocks is completed.

Since we allow repetitions of variables in the sub-blocks, it is prudent to introduce averaging of the multiple appearances of variables. In the case of Jacobi, this can be done naturally at the end of a sweep. In the case of Gauss-Seidel, one needs to keep a count of the repetitions and perform a running average for each repeated variable.

Parallel asynchronous block nonlinear Gauss-Seidel iteration

The idea of chaotic relaxation for linear systems originated with Rosenfeld in 1967 [211]. Other early actors in this important topic were A. Ostrowski [178] and S. Schechter [221]. Chazan and Miranker published in 1969 a detailed paper [43] describing and formalizing chaotic iterations for the parallel iterative solution of systems of linear equations. This was extended to the nonlinear case in [159, 160].

The purpose of chaotic relaxation is to facilitate the parallel implementation of iterative methods in a multi-processor system or in a network of computers by reducing the amount of communication and synchronization between cooperating processes and by allowing that assigned sub-tasks go unfulfilled. This is achieved by not requiring that the relaxation follow a pre-determined sequence of computations, but rather letting the different processes start their evaluations from a current, centrally managed value of the unknowns.

Baudet [10] defines the class of asynchronous iterative methods and shows that it includes chaotic iterations. Besides the classical Jacobi and Gauss-Seidel approaches he introduces the purely asynchronous method in which, at each iteration within a block, current values are always used. This is a stronger cooperative approach than Gauss-Seidel and he shows in numerical experimentation how it is more efficient with an increasing number of processors.

We paraphrase a somewhat more restricted definition for the case of block nonlinear optimization that concerns us. Although Baudet's method would apply directly to calculating the zeros of $\nabla f(\boldsymbol{x})$, we prefer to describe the method in the optimization context in which it will be used. We use the notation introduced above for our partitioned problem.

Let

$$f(\boldsymbol{x}) = \tfrac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_2^2 = \sum_{k=1}^{K} \tfrac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})^{[k]}\|_2^2 = \sum_{k=1}^{K} f(\boldsymbol{x})^{[k]},$$

where the data are partitioned into K non-overlapping blocks. An asynchronous iteration for calculating $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ starting from the vector $\boldsymbol{x}_0$ is a sequence of vectors $\boldsymbol{x}_k$ with elements defined by

$$x_j^{[k]} = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})^{[k]} \qquad \text{for} \qquad j \in \mathsf{CI}_k$$

subject to

$$x_j^{[k]} = x_j \qquad \text{for} \qquad j \notin \mathsf{CI}_k.$$

The initial vector for the kth minimization is

$$\boldsymbol{x}^{[k]} = (x_1(s_1(k)), \ldots, x_n(s_n(k))),$$

where $S = \{s_1(k), \ldots, s_n(k)\}$ is a sequence of elements in N^n that indicates at which iteration a particular component was last updated. In addition, the following conditions should be satisfied:

$$s_i(k) \leq k - 1 \qquad \text{and} \qquad s_i(k) \to \infty, \text{ as } k \to \infty.$$

These conditions guarantee that all the variables are updated often enough, while the formulation allows for the use of updated subsets of variables "as

they become available." Baudet [10] gives sufficient conditions for the convergence of this type of process for systems of nonlinear equations. The convergence of an asynchronous block Gauss-Seidel process and as a special case, the previously described sequential algorithm, follows from the following theorem proved in [188].

Theorem 98. ***Convergence of the asynchronous BGS L-M iteration.*** *Let the operator*

$$T(\boldsymbol{x}) = \boldsymbol{x} - (J(\boldsymbol{x})^T J(\boldsymbol{x}) + \lambda I)^{-1} J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x})$$

be Lipschitz, with constant γ, and uniformly monotone with constant μ, in a neighborhood of a stationary point of $f(\boldsymbol{x})$, with $2\mu/\gamma^2 > 1$. Then T is vector contracting and the asynchronous Levenberg-Marquardt method for minimizing $\frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_2^2$ is convergent.

As we indicated above, an alternative to this procedure is a chaotic block Jacobi iteration, where all the processes use the same initial vector at the beginning of a sweep, at the end of which the full parameter vector is updated. In general, asynchronous block Gauss-Seidel with running averages is the preferred algorithm, since Jacobi requires a synchronization step at the end of each sweep that creates issues of load balancing.

Chapter 10

Ill-Conditioned Problems

Throughout this book, ill conditioning and rank deficiency have been discussed in several places. For completeness, we will summarize the main ideas now. This chapter therefore surveys some important aspects of LSQ problems with ill-conditioned matrices; for more details and algorithms we refer to [20, 118, 121].

10.1 Characterization

As we have already discussed, the consequence of ill conditioning is perturbation amplification. The basic tool for analyzing ill-conditioned linear least squares problems is the singular value decomposition $A = U\Sigma V^T$. Assume that A has no zero singular values and that the data in the LSQ problem are perturbed, i.e., $\boldsymbol{b} = \bar{\boldsymbol{b}} + \boldsymbol{e}$, where $\bar{\boldsymbol{b}}$ is the exact right-hand side, $\bar{\boldsymbol{x}}$ is the exact solution and the noise vector $\boldsymbol{e} = \sum_{i=1}^{n} (\boldsymbol{u}_i^T \boldsymbol{e})\, \boldsymbol{u}_i$ represents white noise with zero mean due to measurement and/or approximation errors. Then the perturbed solution to the LSQ problem is

$$\boldsymbol{x}^* = \sum_{i=1}^{n} \frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i}\, \boldsymbol{v}_i = \sum_{i=1}^{n} \frac{\boldsymbol{u}_i^T \bar{\boldsymbol{b}}}{\sigma_i}\, \boldsymbol{v}_i + \sum_{i=1}^{n} \frac{\boldsymbol{u}_i^T \boldsymbol{e}}{\sigma_i}\, \boldsymbol{v}_i = \bar{\boldsymbol{x}} + \sum_{i=1}^{n} \frac{\boldsymbol{u}_i^T \boldsymbol{e}}{\sigma_i}\, \boldsymbol{v}_i.$$

Given the statistical assumption on $\boldsymbol{e}$, it follows that $\mathcal{E}\big(|\boldsymbol{u}_i^T \boldsymbol{e}|\big)$ is constant. Hence, the contribution to the LS solution from the noise is amplified more by the smaller singular values. This behavior is particularly pronounced when A has a large condition number $\mathrm{cond}(A) = \sigma_1/\sigma_n$. For these ill-conditioned problems we must consider how to deal with the highly amplified noise in the solution.

Ill-conditioned problems are characterized by their distribution of the singular values. Among the ill-conditioned problems there are two impor-

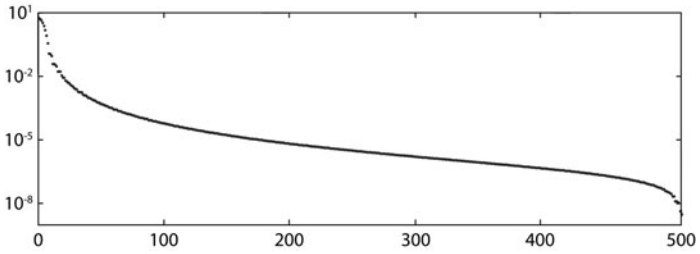

Figure 10.1.1: The singular values of a 512×512 discretization of the "phillips" test problem. Notice that the singular values decay gradually to zero with no specific gap between large and small values.

tant classes that have achieved special attention due to their importance: numerically rank-deficient problems and discrete ill-posed problems.

Numerically rank-deficient problems have a cluster of small singular values with respect to some threshold, and there is a distinct gap between the larger and smaller singular values. The linear prediction problem from Example 28 is an example of such a problem. The numerical rank deficiency underlying this class of problems represents linear dependencies in the underlying mathematical model. The small singular values may be of the order of the machine precision, relative to σ_1, in which case they are nonzero due to rounding errors in the computations. Alternatively, the small singular values can be somewhat larger – but still well separated from the larger ones – in which case their sizes represent approximation and truncation errors in the model.

Discrete ill-posed problems are generated by the discretization of continuous ill-posed problems such as Fredholm equations of the first kind and similar inverse problems. These problems have singular values that decay gradually to zero, and there is no gap anywhere between large and small singular values. This characteristic behavior reflects fundamental properties of the underlying ill-posed problem. Figure 10.1.1 shows the singular values of a discretization of size 512×512 of the particular test problem "phillips" (see p. 32 in [121]). As the matrix dimensions of a discrete ill-posed problem increase, eventually the computed singular values will level off as a result of rounding errors.

10.2 Regularization methods

Whenever small singular values are present, the idea of *regularization* is to extract the linearly independent information from the matrix and the

noisy right-hand side. For numerically rank-deficient problems the concept of numerical rank comes into play, while for discrete ill-posed problems the approach is perhaps less intuitive. In both cases, however, the objective is to suppress the noisy SVD components of the LSQ solution corresponding to the small singular values.

If the problems are "small," in the sense that we can easily compute the SVD, then it is often natural to regularize the problem through the use of the *truncated SVD* (TSVD) method. In TSVD, the number of terms used in the least squares solution is restricted to $k < n$, so that the regularized solution will be

$$\boxed{\text{TSVD solution:} \quad \boldsymbol{x}_k^* = \sum_{i=1}^{k} \frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} \boldsymbol{v}_i.} \tag{10.2.1}$$

A convenient way to interpret this approach is that, instead of solving the original problem with an ill-conditioned matrix, we solve a problem with an approximate, mathematically rank-deficient and better-conditioned one. This is because the TSVD solution $\boldsymbol{x}_k^*$ solves the problem

$$\min_{\boldsymbol{x}} \|A_k \, \boldsymbol{x} - \boldsymbol{b}\|_2 \qquad \text{with} \qquad A_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T. \tag{10.2.2}$$

The regularization is controlled by the truncation parameter k, which is chosen to minimize the influence of noise. It can be shown that the condition number for $\boldsymbol{x}_k^*$ is σ_1/σ_k, which, depending on the choice of k, can be much smaller than $\text{cond}(A) = \sigma_1/\sigma_n$. For numerically rank-deficient problems it is natural to select the parameter k according to the singular values – specifically as the numerical rank with respect to a threshold that reflects the errors in the data. For discrete ill-posed problems, on the other hand, the choice of k must involve the right-hand side; we return to this aspect in the next section.

An alternative strategy is *Tikhonov regularization* [236] (which often gives results that are similar to TSVD, cf. [24], p. 512). Tikhonov regularization can be implemented efficiently for much larger problems because, as we shall see, it leads to a least squares problem. The underlying idea is to add to the LSQ minimization problem a quadratic penalty term that prevents the solution norm from growing large. The original problem is now replaced by computation of the solution $\boldsymbol{x}_\lambda^*$ to the problem

$$\boxed{\text{Tikhonov problem:} \quad \min_{\boldsymbol{x}} \left\{ \|A \, \boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda^2 \, \|\boldsymbol{x}\|_2^2 \right\},} \tag{10.2.3}$$

which is equivalent to the damped least squares problem

$$\min_{\boldsymbol{x}} \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} \boldsymbol{x} - \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix} \right\|_2. \tag{10.2.4}$$

Here λ is the regularization parameter that controls the trade-off between minimizing the residual norm and minimizing the norm of the solution. Since the Tikhonov solution $\boldsymbol{x}_\lambda^*$ is a solution to a LSQ problem, it can be computed by a variety of direct or iterative methods suited for the particular properties of the matrix A (structure, sparsity, etc). A disadvantage is that the damped LSQ problem must be solved for every value of λ, which can be cumbersome if one needs to try many different values; however, if a bidiagonalization of A can be computed (cf. section 5.5), then the computational effort for each λ is reduced considerably.

One often encounters the alternative formulation $(A^TA+\lambda^2 I)\,\boldsymbol{x} = A^T\boldsymbol{b}$, which is the normal equations for (10.2.4) and therefore mathematically equivalent to (10.2.3) and (10.2.4). This formulation is not well suited for numerical computations due to the loss of accuracy when forming the cross-product matrix A^TA with the ill-conditioned matrix A.

A common way to present and analyze the Tikhonov solution $\boldsymbol{x}_\lambda^*$ is via the SVD representation:

$$\boxed{\text{Tikhonov solution:} \quad \boldsymbol{x}_\lambda^* = \sum_{i=1}^{n} \gamma_i \frac{\boldsymbol{u}_i^T\boldsymbol{b}}{\sigma_i}\boldsymbol{v}_i \quad \text{where} \quad \gamma_i = \frac{\sigma_i^2}{\sigma_i^2+\lambda^2}.} \tag{10.2.5}$$

The γ_i are called filters, and they damp the effect of terms corresponding to singular values smaller than λ, thus establishing the relation with the TSVD.

For even larger problems, both the SVD and LSQ computations, associated with the TSVD and Tikhonov methods, are prohibitive. If the matrix A is sparse and one knows beforehand that only a small fraction of the SVD components should be included in the regularized solution, then it could be worthwhile to compute approximate singular values via Golub-Kahan iterations; see [12] and the code LASVD in [263].

An alternative is to use *regularizing iterations*, which rely on the fact that some iterative methods for the LSQ problem, such as CGLS and LSQR, have a regularizing effect, where the number of iterations plays the role of the regularization parameter (see [117] and [121]). Since these methods are defined by Krylov iterations, they restrict the solution $\boldsymbol{x}^{(k)}$ in the kth iteration to a k-dimensional subspace spanned by the right Golub-Kahan vectors. They can therefore be considered as projection methods that reduce the large and ill-conditioned problem to a smaller one with only k unknowns. Using the Courant-Fischer min-max theorem [20] one can show that these approximations are better conditioned than the original problem.

Both theory and computational experience show that the Krylov subspace algorithms in their early iterations produce regularized solutions $\boldsymbol{x}^{(k)}$ that capture the dominating SVD components of the exact solution, i.e., the ones corresponding to the largest singular values. However, at later iter-

ations — as the projected problems converge to the original ill-conditioned one — more noise enters into the solution. That is why the subspace dimension k acts now as the regularization parameter.

In some situations noise may start to dominate the solution $\boldsymbol{x}^{(k)}$ even before all large singular values have been captured by the Golub-Kahan process. When this happens, the "self-regularizing effect" of the projection alone is insufficient to produce a useful regularized solution and instead so-called hybrid methods should be used. Once the dimension of the Krylov subspace is large enough to ensure that we capture all the desired information in the data, a regularization method like TSVD or Tikhonov is introduced for the much smaller bidiagonal matrices of the LSQR method. In this way we still prevent noise from entering the solution; for details see, e.g., [117], [118] and [144]. The disadvantage is that now two parameters have to be chosen: the number of iterations and the regularization parameter for the projected bidiagonal problem.

One important fact in the case of TSVD and Tikhonov regularization, as well as regularizing iterations, is that the solution and the residual norms are monotone functions of the corresponding regularizing parameters. A larger truncation parameter k, a smaller Tikhonov parameter λ, or a larger number of iterations, correspond to an increase of the solution norm and a decrease in the residual norm. This, as we will see in the next section, is important for several parameter selection techniques.

10.3 Parameter selection techniques

In each of the regularization techniques described above some parameter must be selected: the TSVD truncation parameter k, the Tikhonov regularization parameter λ, or the number of CGLS or LSQR iterations. Here we will briefly survey some computational techniques for choosing these parameters. By $\boldsymbol{x}_{\text{reg}}$ we will denote the regularized solution from any of the above methods. We focus on discrete ill-posed problems, where the choice cannot be based solely on the singular values. These methods have in common the fact that they implicitly incorporate information about both the singular values and the left singular vectors through the use of the residual norm.

When accurate information is available about the variance of the noise, represented by the perturbation vector $\boldsymbol{e}$, the regularization parameter can be chosen using Morozov's *discrepancy principle* [144]. Here the regularization parameter is chosen such that the residual norm for the regularized solution is equal to – or close to – the norm of the noise vector:

$$\|\boldsymbol{b} - A\,\boldsymbol{x}_{\text{reg}}\|_2 \simeq \|\boldsymbol{e}\|_2.$$

This technique of "fitting to the noise level" was already introduced in Sec-

tion 6.4 as a stopping criterion for iterative methods. When used as a parameter-choice method, one should be aware that the regularized solution is quite sensitive to underestimates of $\|\boldsymbol{e}\|_2$ [121].

When no reliable estimate is available for $\|\boldsymbol{e}\|_2$, there are several other possible parameter-choice methods. Among them is the *L-curve* technique, which determines the parameter from a log/log plot of the solution norm versus the residual norm, for different values of the regularizing parameter. In other words, each point on the L-curve is given by

$$(\log \|\boldsymbol{b} - A\,\boldsymbol{x}_{\mathrm{reg}}\|_2 \,,\, \log \|\boldsymbol{x}_{\mathrm{reg}}\|_2) \,,$$

where both norms depend on the regularization parameter. For example, in the case of TSVD the two norms depend on the truncation parameter k as follows:

$$\begin{aligned} \|\boldsymbol{x}_k\|_2^2 &= \sum_{i=1}^{k} \left(\frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} \right)^2 , \\ \|\boldsymbol{b} - A\,\boldsymbol{x}_k\|_2^2 &= \sum_{i=k+1}^{n} \left(\boldsymbol{u}_i^T \boldsymbol{b} \right)^2 , \end{aligned}$$

and it is easy to see that the norm $\|\boldsymbol{x}_k\|_2$ of the TSVD solution increases with k, i.e., when more SVD components are included. At the same time, the norm $\|\boldsymbol{b} - A\,\boldsymbol{x}_k\|_2$ of the corresponding residual vector is seen to decrease with k. For Tikhonov regularization, one can show that $\|\boldsymbol{x}_\lambda\|_2$ increases monotonically when λ decreases, while $\|\boldsymbol{b} - A\,\boldsymbol{x}_\lambda\|_2$ decreases monotonically as λ decreases. See [240, 241] for an interesting theory and algorithm based on bi-objective optimization to solve this problem efficiently. Possibly, using [190, 194] techniques for sampling the Pareto front uniformly could be of help in improving even more an excellent algorithm that has many other important applications.

Further analysis in [118, 121] shows that this curve, in log/log scale, will often have a characteristic L shape as shown in Figure 10.3.1. The idea is now to determine the regularization parameter (e.g., k or λ) corresponding to the "corner" of the L-curve, the intuitive argument being that this choice will balance the residual and solution norms. For more details and algorithms we refer to [118, 120, 121]; the methods of Section 8.5 can also be applied. The technique is known to fail if the solution is very smooth, i.e., if the solution is dominated by very few SVD components [119].

When using iterative algorithms based on Golub-Kahan bidiagonalization, one can use the so-called L-ribbons [33] to estimate the best regularization parameter. These are inexpensive by-products of the G-K process, and an extension of this work is the computation of the curvature ribbon described in [32]. A similar methodology for large-scale problems is discussed in [99].

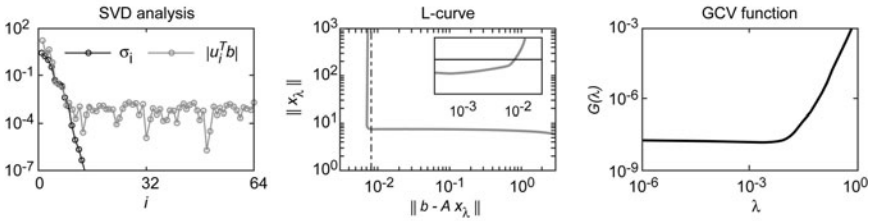

Figure 10.3.1: Shaw test problem. Left: the singular values σ_i and the absolute values of the right-hand side's SVD coefficients $|\boldsymbol{u}_i^T \boldsymbol{b}|$; notice that the latter level off at the standard deviation 10^{-3} of the noise. Middle: the L-curve for Tikhonov regularization with corner for $\lambda = 6.8 \cdot 10^{-4}$; the vertical line at $\|\boldsymbol{b} - A\,\boldsymbol{x}_\lambda\|_2 = \|\boldsymbol{e}\|_2$ represents the discrepancy principle. Right: the GCV function $G(\lambda)$ with a minimum for $\lambda = 8.1 \cdot 10^{-4}$.

Generalized cross-validation (CGV) is another parameter selection technique that does not need additional statistical information about the noise. First proposed by Golub et al. [94], it seeks to minimize the predictive mean-square error $\|\bar{\boldsymbol{b}} - A\,\boldsymbol{x}_\lambda\|_2$, where $\bar{\boldsymbol{b}}$ is the exact right-hand side. The computational problem amounts to minimizing the GCV function, given here for Tikhonov regularization:

$$G(\lambda) = \frac{\frac{1}{m}\|\boldsymbol{b} - A\,\boldsymbol{x}_\lambda\|_2^2}{(\frac{1}{m}\mathrm{trace}(I - F_\lambda))^2}, \qquad F_\lambda = A\,(A^T A + \lambda^2 I)^{-1} A^T. \tag{10.3.1}$$

An efficient algorithm for computing the above GCV function, based on the bidiagonalization of A, can be found in [73].

Yet another approach for choosing the regularization parameter is to perform a residual analysis along the lines described in Section 1.4. An advantage of this approach is that it incorporates statistical information from the right-hand side (instead of just using the residual norm as in the above methods). In the case of white noise in the data, we emphasize the use of the normalized cumulative periodogram as originally advocated by Rust [215, 216], where the key idea is to choose the regularization parameter as soon as the residuals can be considered white noise. For more details and implementation aspects see [122, 216].

A careful comparison of the different parameter-choice methods can be found in [118]. The general conclusion is that the optimal parameter selection technique is problem and regularization method dependent.

Example 99. ***Regularization of the "shaw" test problem.*** *In this example we consider the "shaw" test (see p. 49 in [121]), which models the scattering of light as it passes through a thin slit. We use a discretization*

with $m = n = 64$ and we add white Gaussian noise with standard deviation $\varsigma = 10^{-3}$ to the right-hand side. The left part of Figure 10.3.1 shows the singular values σ_i and the absolute values of the right-hand side's SVD coefficients $|\boldsymbol{u}_i^T \boldsymbol{b}|$; notice that the latter level off at the standard deviation 10^{-3} of the noise. This means that all the SVD coefficients for $i > 10$ are dominated by the noise contributions $\boldsymbol{u}_i^T \boldsymbol{e}$ and therefore should be purged from the regularized solution. This can be achieved by using $k = 10$ in the TSVD method.

The middle part of Figure 10.3.1 shows the L-curve for the problem, and we see two distinct parts – a horizontal part where too few SVD components are included in the solution and a vertical part where too many components are included. In the latter part the noisy SVD components $\boldsymbol{u}_i^T \boldsymbol{e}/\sigma_i$ completely dominate the solution. The corner corresponds to the choice $\lambda = 6.8 \cdot 10^{-4}$ of the Tikhonov regularization parameter. Some of the corresponding filter factors γ_i in (10.2.5) are $\gamma_8 = 0.98$, $\gamma_9 = 0.79$, $\gamma_{10} = 1.2 \cdot 10^{-2}$ and $\gamma_{10} = 2.0 \cdot 10^{-4}$, showing that this choice of λ filters out SVD components for $i > 10$.

The vertical line in the plot at $\|\boldsymbol{b} - A\,\boldsymbol{x}_\lambda\|_2 = \|\boldsymbol{e}\|_2 = 8.2 \cdot 10^{-3}$ represents the discrepancy principle, while the small inset figure shows $\|\boldsymbol{b} - A\,\boldsymbol{x}_\lambda\|_2$ as a function of λ, together with a horizontal line at $\|\boldsymbol{e}\|_2 = 8.2 \cdot 10^{-3}$. For this problem, the discrepancy principle leads to the choice $\lambda = 9.1 \cdot 10^{-3}$, which is somewhat larger than that chosen by the L-curve, leading to a regularized solution with effectively only 8 SVD components.

Finally, the right part of Figure 10.3.1 shows the GCV function $G(\lambda)$ in (10.3.1). The minimum occurs for $\lambda = 8.1 \cdot 10^{-4}$, which is quite close to the value indicated by the L-curve. The figure illustrates a potential problem with the GCV method, namely, that the minimum is quite flat and thus can be difficult to locate.

10.4 Extensions of Tikhonov regularization

The Tikhonov regularization problem in (10.2.3) was motivated by the need to suppress the noisy SVD components of the LSQ solution, and this is achieved by adding the penalization term $\|\boldsymbol{x}\|_2^2$, with weight λ^2, to the LSQ problem. In this section we discuss ways to extend this approach, depending on the user's prior information about the problem.

In some applications we have prior information about the "smoothness" of the solution, in the sense that instead of using the norm $\|\boldsymbol{x}\|_2^2$ as the penalty term we prefer to use $\|L\,\boldsymbol{x}\|_2^2$ where L is a $p \times n$ matrix. If $\mathrm{rank}(L) = n$ then $\|L\,\boldsymbol{x}\|_2$ is a norm, otherwise L has a non-trivial null space and $\|L\,\boldsymbol{x}\|_2$ is a semi-norm; in the latter case, the Tikhonov solution is unique if the null spaces of A and L intersect trivially. Common choices of L are finite-difference approximations to the first or second derivative

since they lead to regularized solutions that tend to be "smoother" than the standard Tikhonov solution $\boldsymbol{x}_\lambda^*$, in the sense that there is a higher correlation between neighboring elements of the solution (and the solutions appear as a smoother function when plotted). If we do not incorporate boundary conditions about the solution, then the standard choices of L are the two rectangular matrices

$$L_1 = \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}, \qquad L_2 = \begin{pmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{pmatrix},$$

which represent discrete approximations to the first- and second-derivative operators on a regular grid. Variants of these matrices that incorporate boundary conditions, and extensions to two-dimensional problems, are discussed in Section 8.2 of [121]. The regularization and parameter-choice algorithms for the case $L = I$ can easily be extended to the case $L \neq I$; we refer to [118] for more details about theory, algorithms, and how to work with semi-norms. See also section 7.1 on the LSQI problem.

The above choice of penalty term is appropriate when the goal is to compute smooth solutions; however, in certain applications we prefer to compute solutions that are less smooth than those produced by Tikhonov regularization. An important case is when our regularized solutions are assumed to be piecewise smooth, i.e., we allow a small amount of elements in the solution that are not correlated with their neighbors, leading to solutions with small "jumps" in the size of the elements. The corresponding penalty term originates in the concept of total variation (which is a fundamental tool in solving nonlinear conservation law problems and in image processing), and it takes the form $\|L_1 \boldsymbol{x}\|_1$, assuming that the solution $\boldsymbol{x}$ represents a one-dimensional signal; see, e.g., [55] for the extension to the two-dimensional case.

Total variation regularization is now used successfully in two- and three-dimensional problems, such as image deblurring and tomographic reconstruction. There are too many different algorithms available today for solving the total variation regularization problem to survey here; most of the algorithms reformulate the problem as a convex optimization problem with a smooth approximation to $\|L_1 \boldsymbol{x}\|_1$ and then use large-scale techniques tailored to the particular application. See [55] for a recent survey of software for total variation image deblurring.

Example 100. *This example illustrates the use of Tikhonov regularization with $L = I$ and $L = L_2$ as well as total variation regularization. The underlying example is the test problem "shaw" from Example 99 with custom-made solutions. The top and bottom rows in Figure 10.4.1 show reconstructions*

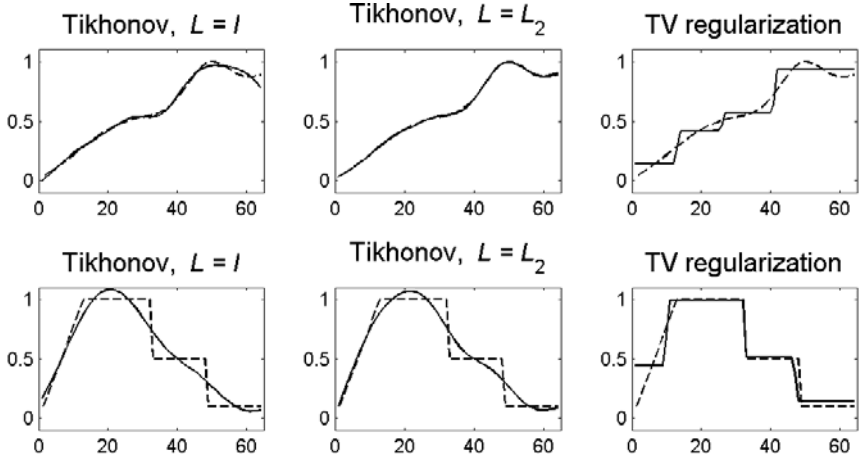

Figure 10.4.1: Reconstructions of smooth and non-smooth solutions by means of Tikhonov and total variation regularization.

(solid lines) of a smooth and a non-smooth solution, respectively (shown as the dashed lines). For the smooth solution, Tikhonov with $L = L_2$ gives the best reconstruction, while for the non-smooth solution the total variation reconstruction is superior.

Prior information can take other forms, and we have already seen, in Chapter 1, that different types of noise in the data lead to different minimization problem. Underlying the total least squares (TLS) problem from Section 7.3 is the fact that the coefficient matrix can also be influenced by errors. We can incorporate this point of view into the Tikhonov formalism and define the *regularized TLS* (R-TLS) problem:

$$\min_{\boldsymbol{x}} \left\{ \frac{\|A\,\boldsymbol{x} - \boldsymbol{b}\|_2^2}{1 + \|\boldsymbol{x}\|_2^2} + \lambda^2 \|L\,\boldsymbol{x}\|_2^2 \right\}, \tag{10.4.1}$$

where the matrix L is as discussed above. We make the following observations related to this problem.

1. The Tikhonov problem (10.2.3) with smoothing term $\|L\,\boldsymbol{x}\|_2^2$ and the R-TLS problem (10.4.1) have the alternative formulations

$$\min_{\boldsymbol{x}} \|A\,\boldsymbol{x} - \boldsymbol{b}\|_2^2 \quad \text{s.t.} \quad \|L\,\boldsymbol{x}\|_2 \leq \alpha$$

and

$$\min_{\boldsymbol{x}} \frac{\|A\,\boldsymbol{x} - \boldsymbol{b}\|_2^2}{1 + \|\boldsymbol{x}\|_2^2} \quad \text{s.t.} \quad \|L\,\boldsymbol{x}\|_2 \leq \alpha$$

where α is a positive parameter. If α is smaller than $\|L\,\boldsymbol{x}^*\|_2$ or $\|L\,\boldsymbol{x}_{\rm TLS}\|_2$, respectively (which is the relevant case for regularization problems), then the solutions to the above two problems satisfy the constraint with equality. We assume this in the discussion below.

2. For the case $L = I$ the R-TLS problem (10.4.1) takes the form $\min_{\boldsymbol{x}}\{(1+\alpha^2)^{-1}\|A\,\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda^2\,\|\boldsymbol{x}\|_2^2\}$, which is identical to the standard-form Tikhonov problem in (10.2.3). Hence in this case the R-TLS solution is identical to the Tikhonov solution. There is no need to solve the more complicated R-TLS problem.

3. For the case $L \neq I$ it was proved in [93] that the R-TLS solution is a solution to the problem

$$(A^TA - \lambda_1 I + \lambda_2 L^TL)\,\boldsymbol{x} = A^T\boldsymbol{b},$$

where the two parameters λ_1 and λ_2 are given by

$$\lambda_1 = \frac{\|A\,\boldsymbol{x} - \boldsymbol{b}\|_2^2}{1 + \|\boldsymbol{x}\|_2^2} \qquad \text{and} \qquad \lambda_2 = \frac{(\boldsymbol{b} - A\,\boldsymbol{x})^T A\,\boldsymbol{x}}{\alpha^2}$$

and it is also shown in [93] that $\lambda_2 > 0$ as long as $\alpha < \|L\,\boldsymbol{x}_{\rm TLS}\|_2$. This means that the R-TLS solution is genuinely different from the Tikhonov solution whenever the residual vector $\boldsymbol{b} - A\,\boldsymbol{x}$ is different from zero, since both λ_1 and λ_2 are nonzero in this case.

The most common algorithms for solving the R-TLS problem are variants of an iterative procedure that require at each iteration the solution of a quadratic eigenvalue problem; see [225] and [207]. A MATLAB implementation RTLSQEP of such an algorithm was written by D. M. Sima and is available from homes.esat.kuleuven.be/~dsima/software/RTLS/RTLS.html. An alternative algorithm described in [11] is based on solving a sequence of quadratic optimization subproblems with a quadric constraint.

10.5 Ill-conditioned NLLSQ problems

An ill-conditioned nonlinear least squares problem has a solution that is very sensitive to data perturbations. The notion of rank deficiency can be extended to the nonlinear least squares problem. We say that a nonlinear problem is rank deficient if the Jacobian $J(\boldsymbol{x})$ has rank $r < \min\{m, n\}$ in a neighborhood of a local minimum of the objective function $f(\boldsymbol{x})$. Of course, a Jacobian that is rank deficient along the iteration also presents a problem (branching or bifurcation), which may call for additional techniques such as continuation to follow different branches leading most likely to different solutions.

Note that all the optimization methods introduced to solve the NLLSQ problem depend heavily on the behavior of the Jacobian, even if the Hessian is considered, since the Jacobian is also the dominant part of the second-order information. One regularization technique, in the rank-deficient case, is the TSVD approach, where one replaces the Jacobian J by J_k, the rank-k approximation (10.2.2) defined using the SVD of J. Unfortunately, the determination of an appropriate truncation parameter k is more costly than in the linear case, since the Jacobian is changing with the iteration.

Tikhonov regularization is another choice. Instead of solving the original minimization problem 8.1.3, the problem one considers is

$$\min_{\boldsymbol{x}} \left\{ \|f(\boldsymbol{x})\|_2^2 + \mu^2 \|\boldsymbol{x} - \boldsymbol{c}\|_2^2 \right\},$$

for some parameter μ. Here $\boldsymbol{c}$ is an approximation to the solution (if one has a good estimate) and otherwise the zero vector.

As we saw in detail in the previous chapter, the method of Levenberg-Marquardt is a regularized version of Gauss-Newton and close in spirit to Tikhonov regularization in the nonlinear case. Böckman [28] has considered regularization via a trust-region algorithm combined with separation of variables and has obtained excellent results in comparison with state-of-the-art solvers applied to the unreduced problem.

Chapter 11

Linear Least Squares Applications

We now present in detail two larger applications of linear least squares for fitting of temperature profiles and geological surface modeling. Both of them use splines, so for completeness, we start with a summary of their definitions and basic properties.

11.1 Splines in approximation

This is a short summary of splines with an emphasis on univariate and bivariate cubic splines, their B-splines representation, and their use in least squares approximation. The topic of splines is important in many data fitting applications, especially where there is no pre-determined functional form. There is a distinct advantage in using splines for data representation, whether in interpolation or least squares approximation, because of their local support and consequent good quality of local approximation. We concentrate in detail on cubic splines because they are well suited to our type of applications. For an exhaustive discussion see some of the classical books, such as [7, 29, 69].

Univariate splines

A spline is a function defined piecewise over its domain by polynomials of a certain degree that are joined together smoothly.

Definition 101. A spline $s^{(k)}(x)$ of degree k, with domain $[a, b]$ and nodes or knots (breakpoints) $a = t_0 \leq t_1 \leq \ldots \leq t_N = b$ has the following properties:

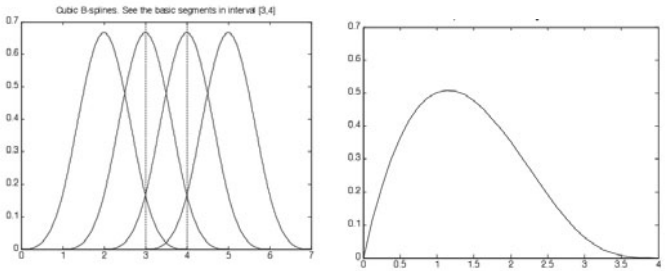

Figure 11.1.1: Left: four consecutive cubic B-splines defined on the uniformly distributed knots $t_j = 0, 2, \ldots, 7$. Right: Cubic spline with coincident end knots at 0.

- *it is a polynomial of degree k in each subinterval $[t_j, t_{j+1}]$, $j = 0, \ldots, N-1$;*
- *at each interior knot $t_1, \ldots, t_{N-1}$ that does not coincide with its neighbors, the function $s^{(k)}(x)$ and its derivatives up to order $k-1$ are continuous.*

The class of such splines is a linear function space of dimension $N + k$, as one can verify from the number of conditions imposed for the continuity of the splines and its derivatives.

There are various possible representations for a spline function, but the numerically most stable is in terms of a basis of *B-splines.*

Definition 102. Given the non-decreasing sequence of knots $\{t_j\}_{j=0,\ldots,N}$, the normalized jth B-spline of degree k is defined by

$$B_j^{(k)}(x) = (t_{j+k+1} - t_j)\, [t_j, \ldots, t_{j+k+1}]\, (t-x)_+^k,$$

where $(t-x)_+^k \equiv \max((t-x)^k, 0)$ is the truncation function and $[t_j, \ldots, t_{j+k+1}]\,(t-x)_+^k$ denotes the kth divided difference of the truncation function with x fixed.

Each $B_j^{(k)}(x)$ is in fact a specific k-degree spline with local support or "active" on $[t_j, t_{j+k+1}]$, i.e., nonzero only in $k+1$ consecutive subintervals. Vice versa, at any $[t_j, t_{j+1}]$, only $k+1$ nonzero B-splines overlap. Also, at any $x \in [t_j, t_{j+1}]$, the sum $\sum_j B_j^{(k)}(x) = 1$. If the knots are uniformly distributed, the maximum value of any B-spline $B_j^{(k)}(x)$ occurs at the midpoint of its "definition" interval. See Figure 11.1.1 for an illustration with $k = 3$.

Instead of deriving the B-splines from the definition, one can generate a B-spline of any degree recursively, starting from the piecewise constant

B-splines $B_j^{(0)}$, by using the de Boor-Cox formulas on the knot sequence $\{t_j\}_{j=0,\ldots,m}$. For $k = 1, 2, ...$

- $B_j^{(0)}(x) = \begin{cases} 1, & t_j \le t < t_{j+1} \\ 0, & \text{otherwise.} \end{cases}$
- $B_j^{(k)}(x) = \frac{x - t_j}{t_{j+k} - t_j} B_j^{(k-1)}(x) + \frac{t_{j+k+1} - x}{t_{j+k+1} - t_{j+1}} B_{j+1}^{(k-1)}(x),$

When some knots coincide, any resulting indeterminate of the form $\frac{0}{0}$ is defined as 0.

The knot distribution is very important, because the relative spacing of the nodes determines the shape of the B-splines. Neither scaling nor translating the knot vector affects the shape. This implies that in the case of uniformly spaced knots (i.e., $t_{j+1} - t_j = \text{constant } \forall j$), these equations can be simplified using the knot vector $\{0, 1, \ldots, N\}$.

We will concentrate now on cubic splines and simplify the notation by omitting the upper-script $k = 3$, i.e., we use $s(x)$ for a cubic spline and $B_j(x)$ for a cubic B-spline. Returning to the representation of splines in terms of B-splines, a general spline can be expressed as

$$s(x) = \sum_j c_j B_j(x), \qquad x \in [a, b].$$

The coefficients c_j are known as *control vertices* or *control points* (from the application of splines in computer graphics).

The knots $\{t_0, t_1, \ldots, t_N\}$ define the $N-3$ splines $B_0, \ldots, B_{N-4}$. Therefore, to have a basis of $N + 3$ functions for the space of cubic splines on the domain $[a, b]$ one needs additional knots to define the 6 additional basis functions required. Thus, 6 more knots, sometimes called *phantom knots,* must be associated with $t_{-3}, t_{-2}, t_{-1}, t_{N+1}, t_{N+2}, t_{N+3}$. These knots can be chosen to coincide with the end points a and b, or to be outside the interval $[a, b]$. The right plot in Figure 11.1.1 illustrates this point for the case $N = 1$ with a single knot interval $[t_0, t_1] = [3, 4]$.

The set $\{B_{-3}, \ldots, B_{N-1}\}$ is now a basis for any spline $s(x)$ defined on $x \in [a, b]$:

$$s(x) = \sum_{j=-3}^{N-1} c_j B_j(x).$$

As mentioned before, the cubic B-splines are only active (i.e., nonzero) in four consecutive intervals, so a change in a control vertex c_j only changes the *local* behavior of the curve.

Multiple knots are particular cases of a nonuniform knot distribution. Another important use of multiple knots is to reduce smoothness. If two knots coincide, the continuity of the derivatives of the spline is reduced by

one order, i.e., for a cubic spline, instead of continuity of derivatives up to order 2, one would have only first-order continuity. In general, there is a reduction of one order of continuity for each additional coincident knot. A special case is *an open uniform* distribution where 3 knot values at each end of the knot vector coincide. This is useful for imposing boundary conditions.

One can use a nonuniform distribution of knots to improve the approximation quality. The difference in the number of knots needed for a given approximation accuracy, be it by interpolation or least squares approximation, can be remarkable, compared to a uniform distribution. For example, when interpolating the function $\sqrt{x}$ in $[0, 1]$ to a $\mathcal{O}(10^{-4})$ precision with a uniform distribution one needs $N > \mathcal{O}(10^6)$, whereas with a nonuniform one it can be done with $N \approx 70$ (see [202], p. 255).

Some of the algorithms that use nonuniform knot sequences may require the addition of knots. It is of interest therefore to have a technique that, given a spline $s(x)$ defined on a knot sequence $\{t_j\}_{j=0,\ldots,m}$, computes the coefficients of the representation of the spline when some knots have been inserted, maybe for added accuracy. An efficient algorithm for this is the Oslo algorithm, described in [69], chapter 1. In the same reference, formulas for the derivative and integral of splines are given, as well as an algorithm for calculating the Fourier coefficients that is of interest in signal processing.

Cubic splines least squares approximation

Originally, splines were designed for interpolation, but this is not appropriate in the presence of noise. In this case we recall that the data are of the form

$$y_i = \Gamma(x_i) + e_i, \qquad i = 1, \ldots, m,$$

with Γ a smooth, low-frequency function, whereas the errors e_i generally consist of all frequencies, including high-frequency oscillations. The "local" character of splines, where the value of any control vertex c_j is only affected by the data in a few neighboring intervals of the knot t_j, can promote the approximation not only of the data trend but also of the less smooth noise, more so than when using more global approximating functions.

Essentially there are two main strategies when using splines to obtain a smooth fit to noisy data:

- smoothing splines,
- regression splines.

For completeness and because of their importance, we briefly describe smoothing splines (following [29]), although we will not pursue them in detail here. Given our fitting problem for the data set $\{(x_i, y_i)\}_{i=1,\ldots,m}$ with $y_i = \Gamma(x_i) + e_i$ and with estimates ς_i of the standard deviation in e_i, a

smoothing cubic spline is constructed by solving the following minimization problem over the space of all cubic splines

$$\min_f \left\{ p \sum_{i=1}^{m} \left[\frac{y_i - f(x_i)}{\varsigma_i} \right]^2 + (1-p) \int_{t_0}^{t_m} f''(x)^2 dx \right\}, \quad 0 < p < 1.$$

The parameter p is chosen to balance the closeness of the fit with the smoothness of the approximating function. This is actually, again, a bi-objective optimization problem and different values of the parameter p will give different points in the Pareto front. Which one to choose is problem dependent, but as usual, the most complete solution would require finding an uniform sampling of the Pareto front and then choosing the appropriate trade-off between accuracy of the fit and smoothness.

Another way to approximate the smooth function contained in the data is via a least squares approximation with splines, a technique known under the name of regression splines. The key is now that the knot sequence is not the set $\{x_i\}_{i=1,\ldots,m}$ at which the data y_i were sampled but an independent sequence covering the same domain interval $[a, b]$ where all the x_i are contained. In general, fewer knots will give smoother splines; the question is to determine the number and distribution of knots to achieve the optimal representation. For a unique solution to exist, every knot interval must contain at least one x_i (adaptation of the Schoenberg-Whitney theorem, see [29]).

Once a set of knots has been chosen, the appropriate spline vertices c_j are determined as solutions of a linear least squares problem:

$$\min_{\boldsymbol{c}} \|\boldsymbol{y} - B\,\boldsymbol{c}\|_2^2,$$

where the so-called observation matrix B is the $m \times (N+3)$ matrix of B-splines functions sampled at the abscissas of the data points

$$b_{ij} = B_j(x_i), \qquad i = 1, \ldots, m, \quad j = 1, \ldots, N+3.$$

For practical reasons we have shifted here the sub-indices, both of the c_j, and the $B_j(x)$ from $-3, \ldots, N-1$ to $1, \ldots, N+3$.

We will assume that the data points x_i are given in increasing order. The matrix B has full rank if, as mentioned before, there are enough data in each knot interval; cf. [29]. If there are not enough data per subinterval, it may still have full rank but be ill-conditioned.

Structurally, B is a sparse matrix with row bandwidth 4 for cubic splines. Let us partition the matrix into $N-1$ blocks,

$$B = \begin{pmatrix} B_1 \\ \vdots \\ B_{N-1} \end{pmatrix} \qquad \text{with} \qquad B_j \in \mathbb{R}^{m_j \times (N+3)}, \quad j = 1, \ldots, N-1,$$

where the number m_j of rows in the jth block is the number of data per subinterval. Then each block is sparse and the four nonzero elements of each row are in columns j through $j+3$ (since in every subinterval only four B-splines are active).

For very large problems, Lawson and Hanson ([150], chapter 27) describe a sequential Householder QR process that is applied in $N-1$ steps and can be tailored to reduce considerably the working storage needed, namely, from$(N+3+\max m_j)\times 5$ to $m\times(N+3)$ locations. At each step, only one new block is incorporated and the already obtained QR decomposition is updated to consider this block. The algorithm requires $m(N+3)^2-(N+3)^3/3$ flops; see the example in [150], p. 221. Alternatively, row-wise Givens rotations can be used efficiently to compute the QR decomposition with low fill-in. A good reference for details regarding adaptations of storage methods and algorithms for normal equations and Householder and Givens QR decompositions to banded matrices is [20].

A considerably more complex problem is the nonlinear least squares problem resulting when the number, but not the distribution, of the knots is chosen a priori. In other words, the minimization problem must be solved for both the knots and the control vertices:

$$\min_{\boldsymbol{c},\boldsymbol{t}} \sum_{i=1}^{m} \left[y_i - s(x_i;\, \boldsymbol{c},\, \boldsymbol{t})\right]^2 .$$

This is a nonlinear constrained separable least squares problem, as the knots have to be in non-descending order. Therefore, the methods described in the previous chapters can be applied. In [137] the difficulties that appear are discussed; see also chapter 4 in [69].

Bivariate tensor product cubic splines

We now describe bivariate tensor product splines; the extension to higher dimensions is straightforward. Some definitions will be needed.

Definition 103. Given the space $\mathcal{U}$ of functions defined on a domain $\mathcal{X}$ and the space $\mathcal{V}$ of functions with domain $\mathcal{Y}$, the tensor product of two functions $u\in\mathcal{U}$ and $v\in\mathcal{V}$, is a function with domain $\mathcal{X}\times\mathcal{Y}=\{(x,y)\,,\ x\in\mathcal{X},\ y\in\mathcal{Y}\}$:

$$u\otimes v\,(x,y) = u(x)v(y).$$

The tensor product of the spaces $\mathcal{U}\otimes\mathcal{V}$ is the set of all such products and their linear combinations.

The Kronecker or tensor product of two matrices $A\in\mathbb{R}^{m\times n}$ and $B\in$

$\mathbb{R}^{p\times q}$ is the $mp \times nq$ block matrix,

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{pmatrix}.$$

An important property is that the pseudoinverse of the Kronecker product of matrices is the Kronecker product of their pseudoinverses: $(A\otimes B)^\dagger = A^\dagger \otimes B^\dagger$. By $\mathrm{vec}(X)$ we denote a one-dimensional (column) array obtained by storing a matrix X consecutively by columns. We are interested in bicubic splines written in terms of a tensor product basis of cubic B-splines.

Definition 104. Consider the rectangular region $[a, b] \times [c, d]$ and define a mesh on it with knots $(t_j^{(x)},\ t_k^{(y)})$ for $j = 0, \dots, J$ *and* $k = 0, \dots, K$. Assume further that $B_j(x)$ and $B_k(y)$ are the cubic B-splines defined as above on the corresponding one-dimensional knot sequences in the x and y directions. Then, provided that (as in 1D) additional knots (phantom knots) have been added, i.e., $(t_j^{(x)},\ t_k^{(y)})$ for $j, k = -3, -2, -1$ and $j = J + 1,\ J + 2,\ J + 3$, $k = K + 1,\ K + 2,\ K + 3$, a basis of bicubic tensor product B-splines is formed by the terms $B_j(x)B_k(y)$.

A general two-dimensional tensor product spline can be written as

$$s(x, y) = \sum_{j=-3}^{J-1} \sum_{k=-3}^{K-1} c_{jk} B_j(x) B_k(y), \qquad \forall (x, y) \in [a, b] \times [c, d].$$

The basis functions have now support in 4×4 adjacent sub-rectangles or cells. And, vice versa, only 16 terms $B_j(x)B_k(y)$ are active in each cell; see Figure 11.1.2 for an example.

The properties of the bicubic spline are analogous to the 1D case: on each sub-rectangle $[t_j^{(x)}, t_{j+1}^{(x)}] \times [t_k^{(y)}, t_{k+1}^{(y)}]$ the spline $s(x, y)$ is a polynomial of degree 3 in x and y. The spline and all its derivatives up to second order are continuous under the provision, as before, that the knots are distinct in both directions:

$$\frac{\partial^{i+l} s(x, y)}{\partial x^i \partial y^l} = \sum_{j=-3}^{J-1} \sum_{k=-3}^{K-1} c_{jk} \left(\frac{\partial^i B_j(x)}{\partial x^i} B_k(y) + B_j(x) \frac{\partial B_k^l(y)}{\partial y^l} \right) \quad i, j = 0, \dots, 2$$

The dimension of the vector space of bicubic splines is $(J+3)(K+3)$. Bivariate splines inherit (with suitable adjustments) the properties described for 1D: smoothness, shape invariance, etc. Details of evaluation, derivation and integration can be found in [69], chapter 2.

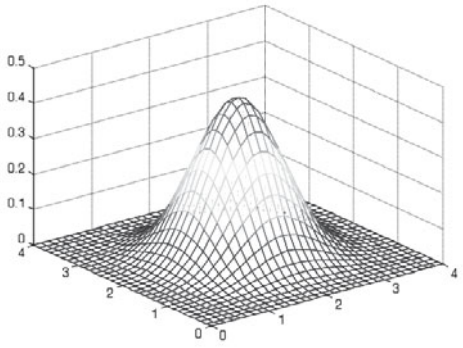

Figure 11.1.2: Bicubic tensor product spline.

Least squares approximation with bicubic splines

We now consider the following problem. Given values of the dependent variable z, corresponding to a set of independent variables (x, y) in the domain $\mathcal{D} = [a, b] \times [c, d]$, find a bicubic spline best l_2-approximation to the values of z. We assume that $\mathcal{D}$ is covered with a rectangular spline knot mesh as described above.

In the two-dimensional approximation problem, the distribution of the data points (x, y) in $\mathcal{D}$ determines whether a number of techniques from one dimension can be carried over.

In many applications one has uniform mesh data, i.e., the values of z are sampled on a rectangular grid $\mathcal{G} = \{(x_i, y_l)\}_{i=1,\ldots,I;\, l=1,\ldots,L}$. In this case, the solution of the two-dimensional problem can be reduced to a cascade of one-dimensional problems. This is possible because, for mesh data, the observation matrix B can be written as the Kronecker product of observation matrices B_x and B_y corresponding to each of the independent variables. The least squares problem is then

$$\sum_{i=1}^{I} \sum_{l=1}^{L} \left(z_{il} - \sum_{j=-3}^{J-1} \sum_{k=-3}^{K-1} c_{jk} B_j(x_i)\, B_k(y_l) \right)^2 ,$$

which in matrix form can be written as

$$\| (B_x \otimes B_y)\, \mathrm{vec}(C) - \mathrm{vec}(Z) \|_2^2 ,$$

where the matrices C and Z contain the control vectices and the data, respectively. Its solution can be written in terms of the pseudoinverse of the tensor product of the matrices. A fast algorithm has been developed by Eric Grosse [112] based on the work in [195].

In the not uncommon situation that some data values z_{il} on the grid are missing, several general techniques still allow the reduction to 1D problems. One method is to complete the grid by linear interpolation using the nearest available grid values. Another successful alternative is an iterative procedure whereby, starting with some initial values for the unknown z_{il}, the least squares algorithm for full grids is applied recursively to fill in the missing values, until the l_2-error at these gridpoints is small compared to the l_2-error on the whole mesh. It can be proved ([69], Chapter 10) that this algorithm converges. Note that the initial values are "smoothed out" by the least squares, so they need not be accurate.

If the data are scattered, i.e., not on a uniform mesh, another approach for the least squares approximation is needed. We assume again that the data are $z_i = \Gamma(x_i, y_i) + e_i$, $i = 1, \ldots, m$ with (x_i, y_i) the independent variables in the domain $\mathcal{D}$ that may be of irregular shape. As above, we superimpose a splines rectangular knot mesh $(t_j^{(x)}, t_k^{(y)})$ on a rectangle containing the data and must now minimize

$$\sum_{i=1}^{m} \left(z_i - \sum_{j=-3}^{J-1} \sum_{k=-3}^{K-1} c_{jk} B_j(x_i) B_k(y_i) \right)^2 .$$

To write this in matrix form, we store the control vertices c_{jk} in the usual form in vec(C), and on row i of the observation matrix A we put the terms $B_j(x_i)B_k(y_i)$ consistent with the ordering of $\boldsymbol{c} = \text{vec}(C)$. As in the one-dimensional case, we shift the indices so that the numeration of j and k starts with 1. The problem is now

$$\min_{\boldsymbol{c}} \|B\,\boldsymbol{c} - \boldsymbol{z}\|_2^2 .$$

The matrix B is sparse with only $4^2 = 16$ non-consecutive entries per row. One can get a compact block banded (4-diagonal) structure if the data are organized by cells and these in turn are stored by varying first in the y direction (see [69], p. 148ff. for details).

If the problems are large, iterative methods are the most efficient. Another approach for very large data sets, common in geological applications (see Section 11.3), is to take advantage of the local character of splines and partition the global least squares problem into smaller subproblems corresponding to a domain decomposition of both the data set and the control vertices. A block iteration is then started, whereby all these local subproblems are solved and the computed local control vertices are assigned as approximations of the global vertices. The block iteration stops when the control vertices converge. Here again, the smoothing effect of the least squares approximation is helpful, and no more than a couple of block iterations are needed for convergence (see Section 6.5).

With scattered data, if the data are poorly distributed, with some cells containing none or few data points, the problem can be singular or ill-conditioned. Unfortunately, as we observed in our applications and as also pointed out in [112], often there is no real gap in the singular values of the observation matrix B and the numerical rank cannot be determined. In this case the regularization methods of Section 10 ought to be used.

One last word about data approximation with spline tensor products. There is no economical way to adapt the spline knot mesh to features that are not in either of the two axis directions. In fact, to improve the representation of such features it is necessary to refine in both directions, thereby generating a two-dimensional fine mesh around it, but which is only needed in a small part of the domain.

Literature and software

From the extensive literature on splines, the classic book by Carl de Boor [29] treats the theory of univariate splines in detail, as well as the topics of tensor product of splines and smoothing splines. The Fortran programs described and used in that book are available from Netlib and include stable evaluation of B-splines, smoothing, least squares and curve approximation. They also form the core of the MATLAB Spline Toolbox. A concise and clear theoretical introduction of splines and the application to interpolation can be found in the book by Powell [202].

Dierckx's monograph [69] covers carefully and exhaustively the most important aspects and algorithms of data fitting with splines in both one- and two-dimensions. It has detailed discussions of least squares methods and smoothing criteria and gives pointers to available software. It also includes a description of a Fortran package FITPACK for data fitting in one- and two-dimensions. The software package can be found in Netlib under the name of DIERCKX. Another good introduction to splines with applications to computer graphics in mind is the book by Bartels et al. [7].

To discuss in more detail the important case of splines with nonuniform knots is beyond the scope of this book. In addition to the information and software available in [69], one software package in the public domain is spline2 [226].

11.2 Global temperatures data fitting

As discussed in Chapter 1, in some data fitting problems there is a well-understood underlying physical process: one knows the form of the functional relationship between dependent and independent variables and just wants to determine some of its parameters. In other cases, either the functional relationship is very complicated and one wants to approximate it by

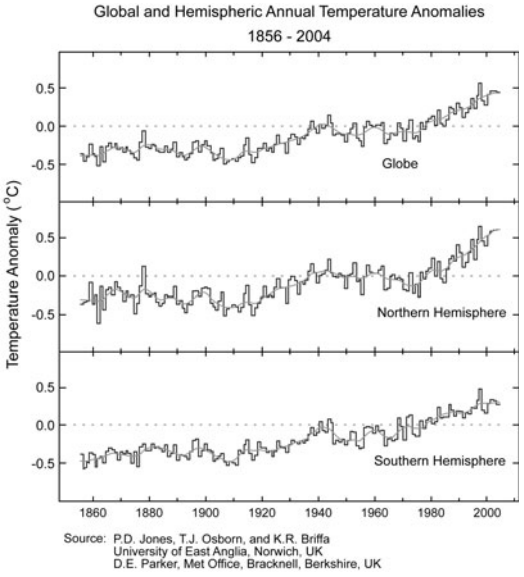

Figure 11.2.1: Earth temperature anomalies (from [135]).

a simpler mathematical function, or one has no previous information about the function type altogether. In these latter cases a parametrized model must be chosen.

We will examine one such problem, describe the steps used to choose a suitable model with the help of some graphical and statistical tools, and then analyze the quality of the computed fit.

Example 105. *The data set under consideration (see Figure 11.2.1), is the time series of the Earth global annual temperature anomalies between 1856 and 1999, compiled at the University of East Anglia [135]. They were obtained from the recorded temperatures by subtracting the global average (14° C) for the years 1961–1990. This set has been exhaustively analyzed by B. Rust and others [213, 222, 234] to try to clarify the question: "Is rising carbon dioxide (CO_2) the cause of the temperature rising?" (See Rust's webpage for interesting background material on climate change [214].)*

How to choose a model to fit a given set of data

We will start with some exploratory data graphics: the scatter plot in Figure 11.2.1 of the temperature T_i versus the time t_i for $i = 1, \ldots, m$ provides insight into the underlying structure of the data and helps decide an appropriate type of approximating function. From this plot of the data set one

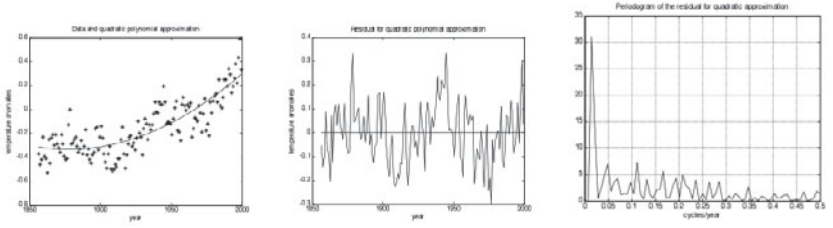

Figure 11.2.2: Quadratic polynomial fit of temperature data, residuals and periodogram.

can appreciate that the relationship is clearly more complicated than linear: there are several max/min (four, at approximately 1870, 1910, 1940 and at 1980), and there seems to be an underlying periodicity. Several data points may be considered as outliers (close to the years 1875, 1940 and 2000).

The model function $M(t)$ should be both simple to evaluate and robust, i.e., well conditioned. Commonly used model functions are polynomials, rational functions, exponential functions, trigonometric functions and their combinations. The simplest choice of approximating function, in order to obtain a linear least squares problem, is a polynomial. Several of these models were explored by Rust [213] and we refer to his careful evaluation of the fit. Here we will only sketch some of the reasons for his choice of model based on residual analysis. A basic quadratic polynomial model is plotted in Figure 11.2.2.

A quadratic will clearly produce an underfit of the data, because from the number of extrema one can deduce that at least a polynomial of degree 5 is needed. Visual inspection of the residual plot in Figure 11.2.2 not only corroborates this, but also shows that the residuals have some systematic variations. Apparently some periodicity of approximately 60–70 years has not been modeled by the polynomial. Figure 11.2.2 also shows a periodogram (or power spectrum, see Section 1.3.3) of the residuals, with a peak indicating a dominant cycle of 72 years. This oscillation was reported on an article in Nature [222], and it was thought to arise from ocean-atmospheric interaction.

Table 11.1 lists the coefficient of determination R^2, defined in (2.2.7), for different models tested in [213]. We see that a model that adds a sine function to the basic quadratic model

$$M(\boldsymbol{c}, t) = c_1 + c_2(t - t_0)^2 + c_3 \sin\left[\frac{2\pi}{c_4}(t + c_5)\right], \tag{11.2.1}$$

explains 80% of the variance in the data, and therefore it provides a good fit.

Model	R^2
Linear	0.618
Quadratic	0.713
Quintic	0.770
Exponential	0.724
Linear + sine	0.758
Quadratic + sine	0.810
Exponential + sine	0.810
Uniform spline ($N = 8$)	0.801

Table 11.1: Coefficients of determination (from [213]).

The disadvantage of model (11.2.1) is that to compute its coefficients one has to solve a nonlinear least squares problem, although for this specific model it can be done using the techniques for separable variables described previously. We will instead try another approach leading to a linear least squares problem.

Since there is no a priori information about the function type we can select a cubic B-spline, a flexible function that can represent almost any structure. As we discussed in the previous section, these functions have the advantage of being well-conditioned, as long as there are enough data in every segment.

The n-term spline is written as $B(t) = \sum_{j=1}^{n} c_j B_j(t)$ *where, as in the previous section, the indices have been shifted to start from* $j = 1$ *and for simplicity, we denote* $n \equiv N + 3$*, where* N *is the number of knot intervals. The spline is defined on a uniformly distributed knot mesh that covers the time interval* $[1856, 1999]$*. With* $n \ll m$*, the approximating spline function can now be determined by solving the linear least squares problem*

$$\min_{\boldsymbol{c}} \| \boldsymbol{T} - B\,\boldsymbol{c}\|_2,$$

where the (i, j)*-element of the matrix* B *is* $b_{ij} = B_j(t_i)$*, i.e., the* j*th B-spline basis function evaluated at the data abscissas* t_i*. The vector* $\boldsymbol{T}$ *denotes the temperature anomalies data and* $\boldsymbol{c}$ *is the parameter vector. The appropriate algorithms to solve this problem were described in the previous section. If the knot sequence is chosen so that there are some data* t_i *in every subinterval, the problem is nonsingular.*

The optimum number of knots, or equivalently, the number n *of basis functions* $B_j(t)$*, must be determined. This will be achieved by striking a balance between smaller residuals* $\boldsymbol{r}^* = \boldsymbol{T} - B\,\boldsymbol{c}^*$ *and fewer parameters* $\boldsymbol{c}^*$ *to be determined, where* $\boldsymbol{c}^*$ *is the LSQ solution. As discussed in Chapter 1, the goal is to find a model that represents the information in the data well enough but "leaves out" the noise, avoiding the danger of overfitting*

n	$\|\boldsymbol{r}^*\|_2^2$	$(s^*)^2$	R^2	adj R^2	$\|\varrho\|/T_\varrho$
6	1.59	0.0115	0.782	0.77	3.83
8	1.45	0.0107	0.801	0.79	3.23
10	1.44	0.0107	0.802	0.79	3.23
12	1.45	0.011	0.801	0.78	3.23
14	1.42	0.0109	0.805	0.79	3.01
24	1.21	0.0101	0.835	0.80	1.59

Table 11.2: Results for different number n of B-splines in the fit to the temperature data.

the data.

Starting with $n = 6$ we will solve the least squares problem with an increasing number of knots and use three statistical measures to assess the goodness of the fit to decide on a provisional optimum value of n: the squared residual norm $\|\boldsymbol{r}^*\|_2^2$, the squared standard error $(s^*)^2 = \|\boldsymbol{r}^*\|_2^2/(m-n)$ (i.e., the squared residual norm adjusted by the residual degrees of freedom, cf. equation (2.2.6)) and the adjusted coefficient of determination adj R^2 *(see 2.2):*

$$\text{adj}\, R^2 = 1 - \frac{\sum_{i=1}^m r_i^2/\,(m-n)}{\sum_{i=1}^m (T_i - \bar{T})^2/(m-1)} = 1 - \frac{(s^*)^2}{\sum_{i=1}^m (T_i - \bar{T})^2/(m-1)}, \tag{11.2.2}$$

where $\bar{T}$ is the mean of the data $\{T_i\}_{i=1,\ldots,m}$. These quantities are shown as a function of n in Table 11.2. Once a tentative "best" value has been selected, we will analyze the corresponding residual vector to evaluate the quality of the approximation.

Under the assumption that the approximation errors are small, i.e., the model represents the pure-data function well, cf. Section 2.2, the squared standard error $(s^*)^2$ is a good estimate of the variance of the noise. The values of $(s^*)^2$ are larger than the noise variance when n is small and decrease as n increases, until they stabilize at a fairly constant value. The first corresponding n is then a reasonable choice for the tentative number of terms. Of course, a small value for $(s^*)^2$ is desirable.

The inconvenience in using R^2 is that increasing the number of coefficients increases R^2 without necessarily improving the fit; this is avoided by adjusting its value with the degrees of freedom. The values of adj R^2 *are smaller than* 1, *and again, a value closer to* 1 *indicates a better fit. Negative values may indicate an overfit. Its use is recommended over the plain R^2 when comparing a series of models obtained by adding parameters.*

Table 11.2 also shows a trend parameter $|\varrho|/T_\varrho$, *appropiate for equispaced data, where ρ is the autocorrelation and T_ϱ is a threshold. (Both are defined and used in section 1.3.) The values of this trend parameter, all*

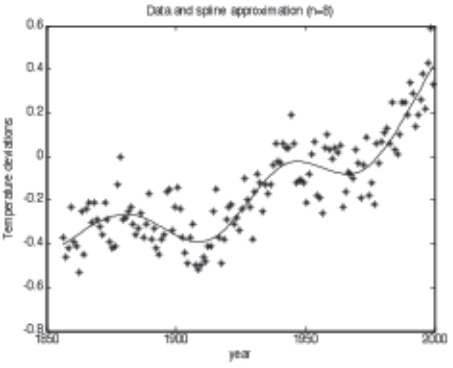

Figure 11.2.3: Cubic B-spline approximation to temperature anomaly data for $n = 8$.

larger than 1, *seem to indicate that there is still structure in the data that has not been modeled adequately. Larger number of knots will of course lead to better approximations of the data, but the work involved in choosing so many parameters appears to be unjustified, although the condition number of the matrix for all the above* n *is not larger than* 40. *The most balanced choice is probably* $n = 8$, *giving the plot shown in Figure 11.2.3. The noise variance estimate* $(s^*)^2$ *has stabilized and is roughly the same for all* n *in the interval* $[6, 24]$ *and, from the values of* $\text{adj}\, R^2$, *the model explains approximately 80% of the data.*

Model validation

As was pointed out already in Section 1.3, in a good fitting model the residuals are practically dominated by the errors in the data and have the same statistical characteristics. Therefore the residuals must be random, uncorrelated, normally distributed with mean zero and identical variance. The quantitative checks of these properties have been described in Section 1.3; hence we take here a complementary approach and concentrate on the use of graphical tools. We want an answer to the following questions for the "best" $n = 8$ *spline function (shown in Figure 11.2.3):*

1. *Are the residuals small?*

2. *Are the residuals random and uncorrelated?*

3. *Is the variation in the response different along the independent variable range?*

4. *Are the residuals (approximately) normally distributed with zero mean and constant variance?*

5. *Is the fit adequate, or do the residuals suggest a better fit, additional terms, or a trend?*

6. *Or, have we overfitted, i.e., can some terms be eliminated?*

In fact, some of the questions are related: if the fit is adequate, the residuals follow the characteristics of the errors, and these in most cases have a normal distribution and vice versa. But looking at each point will allow us to display several of the common techniques for residual analysis.

The first question has been answered numerically, and Figure 11.2.3 confirms it. For the second question, although a plot of the residual versus the independent variable would probably suffice, a confirmation can be obtained from an autocorrelation plot. The vertical axis of this plot is defined by the autocorrelation coefficient R_h with time lag h:

$$R_h = \frac{\frac{1}{m}\sum_{i=1}^{m-h}(r_i - \bar{r})(r_{i+h} - \bar{r})}{\frac{1}{m}\sum_{i=1}^{m}(r_i - \bar{r})^2}, \qquad h = 1, 2, \ldots,$$

where $\bar{r}$ is the mean of the residuals and the horizontal axis is the time lag h. (As this is a plot of residuals, we can assume $\bar{r} = 0$.) The unit-lag autocorrelation ρ in (1.4.3) is a special case for $h = 1$. Autocorrelations should be near zero for randomness. In fact to detect non-randomness it is usually enough to know the unit-lag autocorrelation ρ, i.e., the autocorrelation between r_i and r_{i+1}, which can easily be seen from a lag plot where the vertical axis is r_{i+1} and the horizontal axis is r_i; see Figure 11.2.4. This plot shows no apparent structure and the residuals seem random.

One very important point is to check whether there is a constant variation of the residual (dependent variable) over the course of the observations, because if there is, the standard least squares approximation would not provide good estimates of the parameters, as shown in the example of Chapter 1; instead a weighted approximation is necessary. A plot of the residual can be used to show whether the variation along the range of the independent variable is fixed, and therefore it is unnecessary to use weighted least squares; this is clearly apparent here.

For the fourth question about the distribution of the residuals, a histogram (with vertical axis: counts and horizontal axis: residual values) will give a first idea, and a normal probability plot of the residuals would be further confirmation. Both plots are shown in Figure 11.2.5.

For the normal probability plot, we choose as vertical axis the ordered residuals r_i and as horizontal axis the corresponding normal order statistic medians x_i. These x_i for $i = 1, \ldots, m$ are calculated using the following formula with the inverse of the normal distribution function:

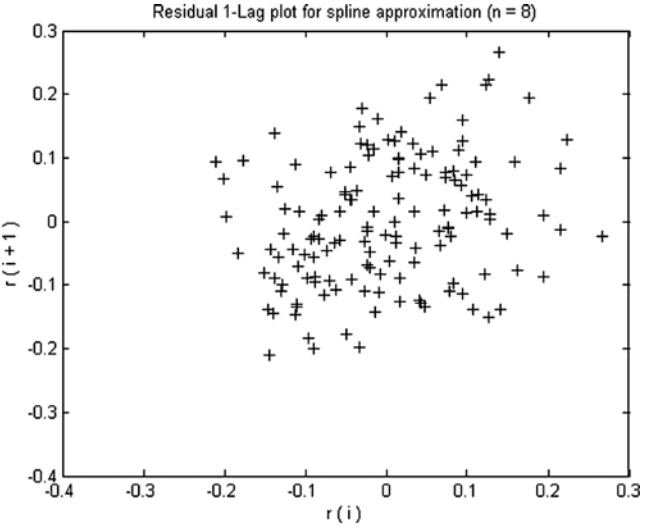

Figure 11.2.4: Unit-lag plot of residuals for the spline approximation with $n = 8$.

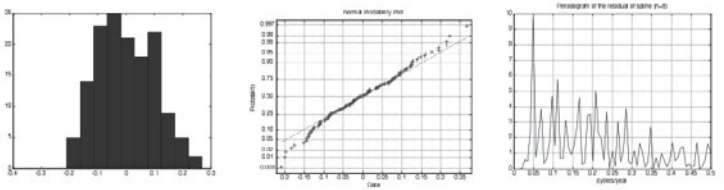

Figure 11.2.5: Statistical plots associated with the spline approximation for $n = 8$. Left: histogram of residual. Middle: normal probability plot. Right: periodogram of the residual.

j	c_j^*	$\pm\delta_j$	c_j^*/δ_j
1	−0.685	±0.725	−0.95
2	0.402	±0.124	−3.2
3	−0.164	±0.063	−2.6
4	−0.566	±0.050	−11
5	0.156	±0.050	3.1
6	−0.260	±0.063	−4.2
7	0.479	±0.124	3.9
8	0.902	±0.725	1.2

Table 11.3: Parameter values and uncertainties for $n = 8$.

$$Pr(X < x_i) = \begin{cases} 1 - 0.5^{1/m} & \text{for } i = 1 \\ 0.5^{1/m} & \text{for } i = m \\ \frac{i-0.3175}{m+0.365} & \text{otherwise.} \end{cases}$$

A way to think about this is that the sample values are plotted against what we would expect to see if the residual distribution were strictly consistent with the normal distribution. If the data are consistent with a sample from a normal distribution, the points should lie close to a straight line.

The histogram, while roughly following the normal distribution bell shape, is skewed with the center displaced. The normal probability plot shows basically a linear pattern except for some deviations at both ends, so on this account the model is not yet very good.

The periodogram of the residual, shown in the right plot of Figure 11.2.5, also points to some underfitting. There is a peak corresponding to a weak periodic component of 20 years per cycle that has not been modeled yet. This is also mentioned in [216], where they use splines with a nonuniform knot distribution. Apparently it corresponds to some real phenomenon, although a physical explanation has not yet been given.

When B has full rank and the noise in the data has covariance matrix $\varsigma^2 I$, the covariance $\mathrm{Cov}(\boldsymbol{c}^*) = \varsigma^2(B^T B)^{-1}$ for the LSQ solution defines the uncertainties in the parameters. If the error variance ς^2 is unknown, as is the case here, an estimated value is given by the above-defined $(s^*)^2$. Let δ_j denote the square roots of the diagonal elements of $\mathrm{Cov}(\boldsymbol{c}^*)$, i.e.,

$$\delta_j^2 = [\mathrm{Cov}(\boldsymbol{c}^*)]_{jj}, \qquad j = 1, \ldots, n.$$

Then δ_j is the standard deviation for the corresponding parameter c_j^*, so that a one-standard deviation interval around this parameter is $c_j^* \pm \delta_j$.

In Table 11.3, the values of the parameters c_j^* with their uncertainties of $\pm 1\delta_j$ are listed. In the rightmost column we list the ratios c_j^*/δ_j of the

parameters to their corresponding uncertainties. If this ratio is large, the probability is small that the parameter c_j^* is zero, making B_j redundant. On the other hand, if the uncertainty interval for a parameter is such that c_j^* may be zero, it is convenient to compute the confidence levels of the parameters. The confidence bounds for c_j^* are given by

$$\Pr\left\{c_j^* - \kappa_p \delta_j < c_j^* < c_j^* + \kappa_p \delta_j\right\} = 1 - p,$$

with $1 - p$ the desired confidence level (a common choice, for example, is $p = 0.05$, for a 95% confidence level). The parameter κ_p depends on the confidence level and is obtained from the inverse of the Student's t-cumulative distribution function with $m - n$ degrees of freedom.

For the above problem, the first and last parameters c_1 and c_8 have a small ratio c_j^*/δ_j, so it is worthwhile to compute the confidence bounds for these two parameters. We use the Student's t distribution with $144-8 = 136$ degrees of freedom, giving $\kappa_p = 0.67449$, and for the first parameter we must evaluate

$$\Pr\{-0.685 - 0.67449 \cdot 0.725 < c_1^* < -0.685 + 0.67449 \cdot 0.725\}$$

and similarly for the last parameter. We obtain

$$\Pr\{-1.174 < c_1^* < -0.1964\} = 0.5,$$
$$\Pr\{0.06866 < c_8^* < 1.736\} = 0.75.$$

Hence, for c_1^* we can only guarantee with 50% probability that the confidence interval does not include zero, while for c_1^* this confidence goes up to 75%. A similar technique can be used to check for overfitting; for the present model it is redundant, as one can rather make a case for underfitting against overfitting.

11.3 Geological surface modeling

An interesting application area, both for linear and nonlinear least squares problems, is exploration seismology, as used in oil and gas prospecting (for details see [224]). One of the problems in this area is the determination of the location and shape of geological formations in the Earth from measurements on its surface. Although the presence of oil or gas per se cannot be detected by the elastic properties of rocks or by the shape of the interfaces, such as where the rocks tilt upward, or where the strata are broken by faults, those are valuable information for identifying favorable conditions of hydrocarbon presence (traps for the fluids that tend to rise and accumulate at plugs provided by impervious rocks). Also, determining the elastic properties of rocks helps, through rock physics, to obtain valuable information

about properties that are useful to reservoir engineering, such as density, porosity and permeability. We will discuss some of these applications in the next chapter, as they lead to nonlinear least squares problems.

Example 106. ***Salt SEG surface***

Our next example will consider the computation of an analytical approximation of a complex geological interface defined by a discrete scattered set of depth data points [58, 197]. This is just one surface from a data set (Salt SEG) created under the sponsorship of the Society of Exploration Geophysicists (SEG). The surface represents the depth of a boundary between two different sedimentary rock formations pierced by a salt body (center hole). The surface includes a large normal fault that runs across the salt, from northwest to southeast, and some smaller faults that produce sharp gradients in the surface.

The set contains $m = 71,952$ *depth data points* (x_i, y_i, z_i) *in the domain* $(x_i, y_i) \in [0, 280] \times [0, 280]$*, with a depth range:* $z_i \in [400, 17300]$*. Although the* (x_i, y_i) *points all lie on a rectangular grid, there are large data gaps in the domain due to the salt body intrusion and several faults (that belong to other surfaces). The depth values* z_i *have not only a wide range but, due to the faults, they present large discontinuities. Figure 11.3.1 shows a plot of the data.*

We assume that the data values can be fitted by a smooth, twice differentiable function, the errors being random and normally distributed. The idea is not to extrapolate to cover the holes, but rather to approximate the data only where the surface is defined. A much more laborious alternative is to segment the domain and approximate the surface in patches that then need to be stitched together.

A suitable mathematical model $M(x, y)$ *to represent this surface is a tensor product of cubic B-splines, having the appropriate smoothness and, an important feature in this case of large domain gaps, having a local support basis functions representation.*

To define the model function we introduce a uniform mesh, determined on the given $[0, 280] \times [0, 280]$ *rectangular domain by the knots:* $(x_j^{\text{knot}}, y_k^{\text{knot}})$ *for* $j = 1, \ldots, J$ *and* $k = 1, \ldots, K$*. The model function is then*

$$M(x, y) = \sum_{j=1}^{J} \sum_{k=1}^{K} c_{jk}\, B_j(x) B_k(y).$$

The coefficients (or control vertices) c_{jk} *are determined by a large, linear, overdetermined and inconsistent system of equations:*

$$\sum_{j=1}^{J} \sum_{k=1}^{K} c_{jk}\, B_j(x_i) B_k(y_i) \approx z_i, \qquad i = 1, \ldots, m.$$

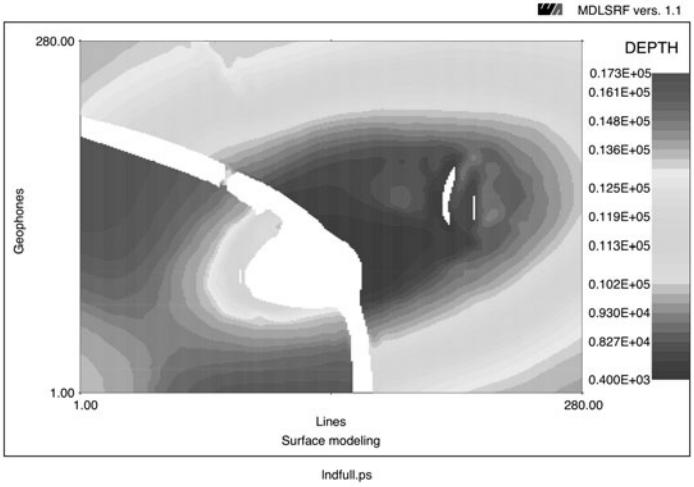

Figure 11.3.1: The data that define the salt SEG surface.

A best fit in the l_2-norm is obtained by solving

$$\min_{c_{jk}} \sum_{i=1}^{m} \left(z_i - \sum_{j=1}^{J} \sum_{k=1}^{K} c_{jk}\, B_j(x_i) B_k(y_i) \right)^2 ,$$

where we assume that $n = J \cdot K \ll m$. As explained in a previous section, the least squares problem can be recast in matrix form $\min_{\boldsymbol{c}} \|\boldsymbol{z} - B\,\boldsymbol{c}\|_2^2$, where $\boldsymbol{z}$ is the vector of the data, the coefficients are stored in the vector $\boldsymbol{c}$ and B is the usual matrix associated with the B-spline basis functions, with the terms $B_j(x_i)B_k(y_i)$ stored in row i consistently with the ordering in the coefficient vector, as explained in detail in the section on splines.

We recall that the problem is sparse, because the B-spline functions have local support – at any given evaluation point only four coefficients (control vertices) in each coordinate are different from zero, for a total of 16 coefficients in two-dimensions. Because of the data "gaps" and the large discontinuities, the problem is ill-conditioned. In fact, if the spline mesh is so fine that a control knot $(x_j^{\text{knot}}, y_k^{\text{knot}})$ is more than two mesh intervals from every data point, the associated coefficient c_{jk} will not enter into any evaluation of the least squares matrix and thus will produce a column of zeros in B. In addition, control vertices associated with knots that are close to only a few data points can be poorly defined by the data and also lead to small singular values and ill-conditioning.

Figure 11.3.2 shows a log-scale plot of the singular value distribution of the matrix B for a 20×20 B-splines grid model of a representative subset

of the data. The singular values are normalized with respect to the largest σ_1. *Note the gap before the last three singular values, which are of order of* 10^{-7} *relative to the largest singular value and are nonzero due to rounding errors. The gap at* 10^{-4} *shows a typical behavior for rank-deficient problems and also suggest the numerical rank of* B.

For the special case that the data are uniformly spaced, a fast algorithm to solve the tensor product least squares problem has been developed [187, 195], based on the reduction to a cascade of 1D problems (tensor product fitting to tensor product data). In the present case, in order to use this reduction, one could re-grid the data to a uniform mesh, filling any gaps by interpolation or extrapolation.

The alternative that we present here instead involves considering the multidimensional least squares problem as a whole. To give a reasonable fit, the number of unknown parameters must be at least $n = 400$. *So we are dealing with a large, linear, sparse and ill-conditioned problem, with the main difficulty being the ill-conditioning.*

The size of the problem excludes the use of a simple truncated SVD algorithm, which otherwise would be a good choice given the rank deficiency. The matrix B *is sparse (16 entries per row), so LASVD, the method developed by Berry, seems an option. Unfortunately, the exploratory SVD computation using a data subset (shown in Figure 11.3.2), indicates that most of the singular values are relevant and would have to be computed, making LASVD impractical. We therefore try BTSVD [196, 197], a block algorithm based on TSVD and compare its performance with the iterative Lanczos method LSQR that has good regularizing properties. BTSVD is a hybrid algorithm, based on partitioning the original problem into subproblems small enough to make a truncated SVD algorithm practical.*

Because of the local character of the B-splines, the global LSQ problem subdivides naturally into subproblems, corresponding to a domain decomposition of the independent variables of the data set and the corresponding relevant control vertices. Some natural overlap arises from the phantom vertices, i.e., the control vertices associated with the two outermost basis functions in each direction, which are affected by the data in the neighboring domain.

Thus, the local LSQ problems are loosely coupled and therefore they are suited to a block Gauss-Seidel approach (for details see [196, 197] and 6.5). The blocks are visited sequentially and a master copy of all the control vertices is updated as soon as a block fit has been completed. This updating strategy should achieve a global C^1 *approximation, since we impose appropriate continuity conditions across the block boundaries. Among the several possibilities to determine a value for the shared phantom vertices we choose to assign a dynamic mean value of the corresponding local values.*

BTSVD can be considered a special case of the block Gauss-Seidel it-

eration considered in Section 6.5; it is amenable to parallelization and the parallel version can be asynchronous. The convergence follows from Theorem 98.

An important issue is the local regularization *(see Chapter 10), since a number of the subproblems are rank deficient. For each of these subproblems one has to define a regularization parameter, i.e., the number l of SVD terms used in a regularized solution. Among several options, we choose the L-curve because it does not require additional information about the data, though, as it is costly to compute the optimum regularization parameter, we have implemented a simplified strategy based on the monotone behavior of the solution norms in order to compute a reasonably good approximation of the parameter.*

The plot of the singular values in Figure 11.3.2 can give some idea of the rank of the complete least squares problem, $\text{rank}(A) \approx r_\tau$*, where* r_τ *is the number of normalized singular values bounded below by* $\tau = 10^{-4}$*. We use this value as a starting value for l. From the first tentative TSVD solution* $\boldsymbol{c}_l$ *and its residual* $\boldsymbol{r}_l = \boldsymbol{z} - B\,\boldsymbol{c}_l$*, successive improved approximations of the local control vertices and the residuals are computed until the desired solution quality is obtained. We stop the process when the solution norm starts to increase dramatically (which is the central idea in the L-curve criterion).*

We have found that the most efficient strategy for BTSVD combines some initial LSQR iterations with a switch to a TSVD solver for handling the ill-conditioning and termination issues. The local LSQR iterations are an essential element of the algorithm because they determine a good initial approximation to the phantom vertices used in the TSVD steps, thus speeding up the convergence of the block Gauss-Seidel method.

On the other hand, Krylov subspace methods are self-regularizing; the subspace dimension (= iteration number) is the regularization parameter. As mentioned in Section 6.4, a reliable, automatic regularization is not easily implemented. The LSQR program requires as input the parameter CONLIM, an estimate of the condition of the bidiagonal matrices B_k *(see Section 6.3) and the algorithm will stop at iteration k if:*

$$\text{cond}(B_k) \geq \text{CONLIM}.$$

This heuristic should have a similar effect as TSVD with a normalized singular value threshold of $\tau \simeq 1/\text{CONLIM}$*.*

Table 11.4 illustrates the difficulty of stopping the LSQR iteration if there is no indication of the appropriate CONLIM value. LSQR was stopped using different values, and we recorded the RMS as well as the percentage of data points with residual farther than 1 standard deviation from the mean ("Large residual" in Table 11.4). We also list the relation of the computer time needed by the two algorithms LSQR and BTSVD to obtain the same

quality of results.

The stopping criterion for the Gauss-Seidel sweeps through the blocks is a bound on the residual mean square $\text{RMS} = \|\boldsymbol{r}\|_2/\sqrt{m}$. *The experience with this and other problems is that at most 3 sweeps over all the blocks are needed to attain the desired precision. The choices* $\text{CONLIM} = 10^3$ *and* $\text{CONLIM} = 10^4$ *for LSQR give similar approximation quality, while the solution for* $\text{CONLIM} = 10^3$ *is computed at a fraction of the number of operations needed when using* $\text{CONLIM} = 10^4$. *A second way to introduce regularization into LSQR is available through the parameter ATOL, which can cause iterations to terminate before the norm of the solution becomes too large. This was our approach.*

BTSVD was run using a 5×5 *block domain decomposition, with* 10×10 *local vertices and a common threshold of* $\tau = 10^{-4}$. *An* $\text{RMS} = 8.18 \cdot 10^{-3}$ *was obtained, with 91% of the data having an error less than 0.5% in 2 sweeps. The first sweep used 32* local *LSQR iterations and the second one used the local TSVD. Figures 11.3.3 and 11.3.4 show plots of the componentwise relative errors obtained with the two methods. We see that, even close to the holes and the large-gradient areas the accuracy of the fit is reasonable, but more important, this difficult local features do not contaminate the smoother regions.*

In general, BTSVD is applicable (and has been applied; see [197]) to data in irregularly shaped domains, including holes, provided that these will also be the domains of evaluation of the model functions. Such domains must first be embedded into a rectangular region.

An important question that we have not yet considered is, How do we choose the number of basis functions J *and* K*? The aim is for the resulting representation to preserve the character of the discrete data without introducing extraneous artifacts and/or excessive smoothing. To address this issue we tried a multigrid strategy. Starting from a coarse spline mesh and possibly a subset of the data, we solved the resulting least squares problem. The quality of the fit was checked via the residual norm. Note that in addition, other properties of the model could be evaluated by computing local differences and comparing them with the derivatives of the fitting function:*

$$B_x(x,y) = \sum_{j,k} c_{jk} B_j'(x) B_k(y), \qquad B_y(x,y) = \sum_{j,k} c_{jk} B_j(x) B_k'(y).$$

The norm of the difference between these quantities can be used in conjunction with the function residual to choose an appropriate parameter space (i.e., the values of J *and* K*).*

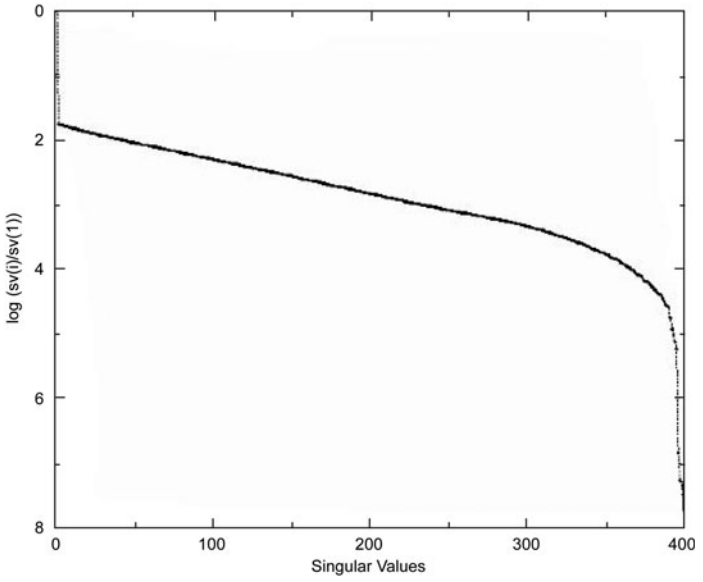

Figure 11.3.2: Singular values for SEG salt model.

CONLIM	RMS	Large res.	# iters	LSQR / BTSVD time
10^2	$8.5 \cdot 10^{-3}$	8%	31	0.1
10^3	$7.1 \cdot 10^{-3}$	5%	66	0.24
$5 \cdot 10^3$	$7.06 \cdot 10^{-3}$	5%	167	0.6
10^4	$7.05 \cdot 10^{-3}$	5%	863	3.1

Table 11.4: LSQR performance for various values of CONLIM.

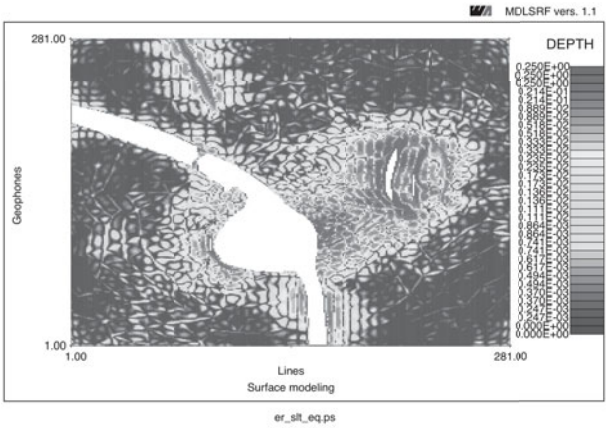

Figure 11.3.3: Relative error for LSQR. As expected, the largest errors (of 25% for about 8% of the points) are close to the faults and the salt inclusion (holes). Elsewhere, the errors are as small as 0.025%.

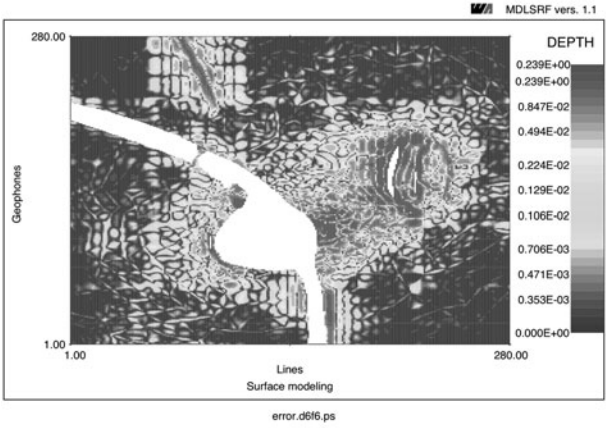

Figure 11.3.4: Relative error for BTSVD. The error behavior is similar to LSQR.

Figure 11.3.5: Monterey Bay, California: Data.

Example 107. ***Monterey Bay, California, topography***

To give an idea of the computing resources needed, we describe a similar exercise with a data set that corresponds to altimetry and bathymetry (height and underwater depth measurements) of Monterey Bay, California, showing the well-known and striking underwater canyon there. The data set has $m = 40,000$ *data points in a uniform* 200×200 *mesh, and the complete approximation uses* $n = 10,000$ *basis functions, for a LSQ problem of dimensions* $40,000 \times 10,000$. *Decomposing the domain into* 5×5 *blocks (of dimension* (40×40)*, initializing this time with the CG method (which gives similar results to LSQR) and then finishing up (because of potential ill-conditioning) with the BTSVD method, we obtain a robust and efficient algorithm that takes less than one minute on a MacBook Pro (with Intel Duo processors).*

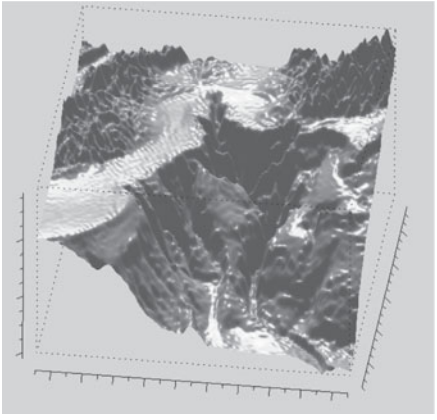

Figure 11.3.6: Monterey Bay, California: B-spline fit.

Chapter 12

Nonlinear Least Squares Applications

In this chapter we consider several nonlinear least squares applications in detail, starting with the fast training of neural networks and their use in optimal design to generate surrogates of very expensive functionals. Then we consider several inverse problems related to the design of piezoelectrical transducers, NMR and geophysical tomography. Not surprisingly, several of the applications lead to separable problems that are solved using variable projection.

12.1 Neural networks training

Neural networks are nonlinear parametric models. Training the network corresponds to fitting the parameters in these models in the least squares sense by using pre-classified data (a training set) and an optimization algorithm. Since the nonlinear least squares problem (NLLSQ) that results is one whose linear and nonlinear variables separate, the use of the variable projection (VP) algorithm is possible, and it increases both the speed and the robustness of the training process [101, 103, 168, 182, 199, 227, 252, 253, 254], compared to traditional methods.

First we explain the necessary neural network concepts and notations that lead to the specific least squares problem that has to be solved to train the network. Then we explain and test a training algorithm based on variable projection. The algorithm is applicable to one hidden layer, fully connected, neural network models using several types of activation functions. A generalization of VP developed and implemented by Golub and Leveque [98] (see also [84, 85, 141]) can be used for the case of multiple outputs.

Neural network concepts

Neural networks are a convenient way to represent general nonlinear mappings between multidimensional spaces in terms of superposition of nonlinear functions, by using so-called activation functions an hidden units.

We discuss here multilayer, feed-forward, fully connected neural networks (NNs), although the techniques are applicable to more general ones, for instance, those with feedback loops. The NN we consider consist of 3 types of *nodes* arranged in *layers*: *input, hidden* and *output* layers. Each node can have several inputs and outputs, and they act on the input information in ways that depend on the layer type:

- Input node: no action.
- Hidden node:
  - Weighted sum of its inputs with a possible offset (bias) that can be incorporated as an additional input. If w_i are the weights and x_i the inputs, with x_0 corresponding to the bias, i.e., $x_0 = 1$, then
$$y = \sum_{i=0}^{d} w_i x_i \equiv \boldsymbol{w}^T \boldsymbol{x}.$$
  - This linear output can be generalized by applying a nonlinear function $f(\cdot)$, called an *activation function*, so that the output is instead $z = f(y) = f(\boldsymbol{w}^T \boldsymbol{x})$. Observe that this provides a standardized way to handle multivariable inputs.
- Output node: it can weight and sum inputs and additionally apply an activation function.

In a feed-forward NN there is information flowing in only one direction. A fully connected NN has every node in one layer connected to every node in the next layer, and there are no connections between nodes of the same layer. See Figure 12.1.1 for the general architecture of a NN.

The notation used is as follows:

- The training set is defined by (input/output) pairs, $\boldsymbol{x}_{1,i}$ and $\boldsymbol{t}_i$, $i = 1, \ldots, m$.
- At the lth-layer, $\boldsymbol{x}_{l,i}$ and $\boldsymbol{x}_{l+1,i}$ denote the input and output vectors, whereas $\boldsymbol{y}_{l,i}$ is used for the vector of weighted sums.
- The final output is $\boldsymbol{x}_{L+1,i}$.
- Note that these vectors may have different lengths in different layers, given that the number of nodes can vary.

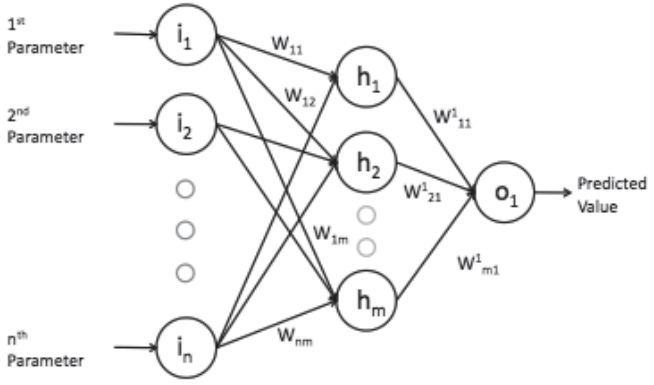

Figure 12.1.1: Neural network architecture.

- At the lth-layer, the weights, for calculating the weighted sums are denoted by the vector $\boldsymbol{w}_l$. The number of its elements is the product of the number of nodes in layer $l-1$, times the number of nodes in layer l.
- The total weight vector for the NN will be denoted by

$$\mathbf{W}^T = (\ \mathbf{w_1}^T \quad \mathbf{w_2}^T \quad \ldots \quad \mathbf{w_L}^T\ ).$$

Perceptron models

These are networks that use nonlinear activation functions, and they are usually applied to classification problems. To allow for a general mapping, one must consider successive transformations corresponding to several layers of adaptive parameters. The most common choice for the activation function is the logistic sigmoid function, see Figure 12.1.2, defined by

$$f(y) = \frac{1}{1+e^{-y}}. \tag{12.1.1}$$

Some of its properties are

- For $y \in (-\infty, \infty)\,,\ f(y) \in (0,1)$.
- The derivative approaches 0 when $|y| \gg 1$.
- It has a simple derivative form: $f'(y) = f(y)(1-f(y))$.
- It is continuously differentiable and able to approximate the hard delimiter step function.

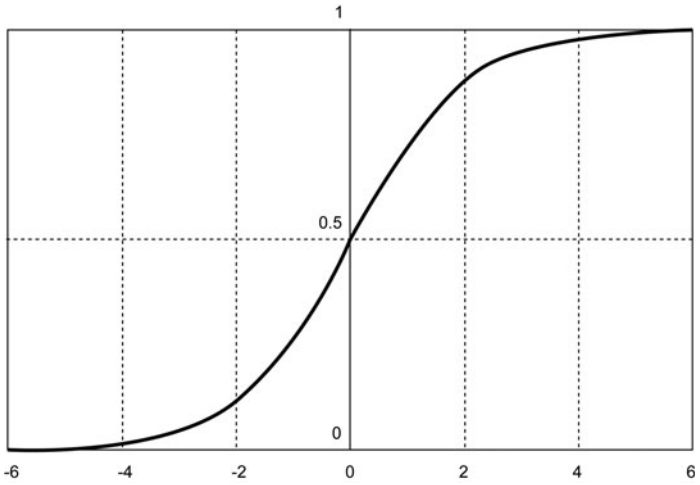

Figure 12.1.2: Sigmoid function.

Since the sigmoid functions are dense in the space of continuous functions, a single hidden layer NN can approximate any continuous function to any precision, if enough nodes are used [54].

Training as a separable nonlinear least squares problem

The training of a neural network uses known input-output pairs (the training data) in order to find the NN parameters that best reproduce their behavior. A frequently used measure of performance is the sum of squares of the residuals between the NN output and that of the training set. Given a training set defined by $(\boldsymbol{x}_{1,i}, \boldsymbol{t}_i)$, $i = 1, \ldots, m$ and assuming that the outputs are independent, the weight parameters $\mathbf{W}$ are to be determined so that the sum of squares residuals is minimized:

$$\min_{\mathbf{W}} \frac{1}{2} \sum_{i=1}^{m} \|\boldsymbol{x}_{L+1,i} - \boldsymbol{t}_i\|_2^2, \tag{12.1.2}$$

where, of course, $\boldsymbol{x}_{L+1,i}$ depends nonlinearly on both the weights $\mathbf{W}$ and the inputs $\boldsymbol{x}_{1,i}$.

When there is a correlation between the outputs, or the errors have different sizes, [181] suggests a more appropiate formulation involving the output error correlation matrix, giving rise to a generalized least squares problem [20]. See also Chapter 6 in [15].

In what follows we will consider a simplified 3-layer NN such that there is only one output node to which no activation function is applied. In this case

the approximating function reduces to a weighted combination of activation functions and the training of the network leads to a separable nonlinear least squares problem, for whose solution we propose to use variable projection.

Why variable projection?

Besides the obvious advantage of reducing the number of parameters to be determined by eliminating the linear parameters, it has been shown [169, 181, 212, 227] that the resulting reduced problem is better conditioned than the original full one and if the same optimization algorithm is used it always converges in fewer iterations. Since with a careful implementation the cost per iteration is about the same, these results and extensive practical experience as reported in [103] (where a wealth of references can be found), show that there is a net gain in using variable projection over solving the unreduced problem with a conventional nonlinear least squares algorithm. Since neural networks are usually trained using very slow algorithms (requiring hundred of thousands of iterations), such as back propagation (gradient method with step control), the gain compared to these conventional training methods is even larger.

On the negative side (for every algorithm), if there are no good a priori initial values for the nonlinear parameters then, these nonlinear, non-convex problems will frequently have multiple solutions and therefore some kind of global optimization technique will need to be used to escape from undesired local minima. We show in the computational section below that a simple Monte Carlo method in which multiple initial values are chosen at random helps with this problem.

Valid concerns of NN practitioners are the design of the network and the so called bias-variance dilemma. Since we are considering single hidden layer perceptrons, the only design parameter in this case is the number of nodes in the hidden layer, which should be as small as possible, since parsimonious models (i.e., models with as few parameters as possible) are to be desired. The techniques we have described before for choosing the number of basis functions in a model are applicable here.

The second concern is related to the fact that training data in real situations will have errors and that the resulting problems are often ill-conditioned and hence there is a real possibility of overfitting the training data. Minimizing the bias (i.e., the residual norm) and the variance are desirable but conflicting goals. As in any bi-objective optimization problem, a compromise has to be reached. The procedure described below uses a modern implementation of the Levenberg-Marquardt method (cf. [151]), which has a built-in approach to palliate the above problem, if an approximation to the variance is available. We have also discussed in detail ill-conditioned problems and regularization in Section 10.

VARPRO program and numerical tests

One of the variable projection implementations we use is based on the original program written by Pereyra [101] as modified by J. Bolstad [263]. The minimization method used is a modification by Osborne of the Levenberg-Marquardt (L-M) algorithm, and it also includes the Kaufman simplification. Careful implementation of the linear algebra involved in the differentiation of the variable projector and the L-M algorithm produces an efficient algorithm. The information provided by the program allows for a statistical analysis, including, for example, uncertainty bounds in the parameter estimations (see [113, 213] for a brief overview).

Analysis and experimental results in [217] for fully connected NN using sigmoid activation functions (which have limited discrimination capabilities), suggest that many network training problems are ill-conditioned. One can show that columns of the Jacobian can easily be nearly linearly dependent. As proved in [227], the use of the variable projection method improves the condition of the problem.

The interesting analysis in [76] might be useful to design an even more robust algorithm that would be applicable not only to the original NN problem but also to nonlinear separable problems in general. In fact the authors show theoretically and practically that regularizing the NN nonlinear problem directly (using the Tikhonov approach to compensate for ill-conditioning) and then linearizing using Gauss-Newton gives a problem with a smaller condition number than the usual approach for a nonlinear problem, namely, to linearize and then regularize (Levenberg-Marquardt).

Example 108. *We consider a test problem from a public benchmark, about the prediction of hourly electrical energy consumption in a building (named "a"), based on the date, time of day, outside temperature, outside air humidity, solar radiation and wind speed. The data can be found in [203]. Complete hourly data for four consecutive months is provided for training, and output data for the next two months should be predicted.*

The purpose of this test is to show the performance of the training algorithm and not to design the "optimal" network. The combination of a small network, a large number of data points, and the regularization provided by the use of the variable projection and Levenberg-Marquardt algorithms should guarantee that there will be no overfitting of the training data (see chapter 9 in [15] for a more complete discussion of these issues).

We use a single hidden layer network with 5 nodes, sigmoid activation functions and a constant bias. Since the extrapolation that is required will not overlap in the time of the year with the training data, we ignore the year and month and use only an hour count, hour of the day count, and a day of the week count as time input variables. We also include the output data of the three previous hours as memory (previous hour, first and second

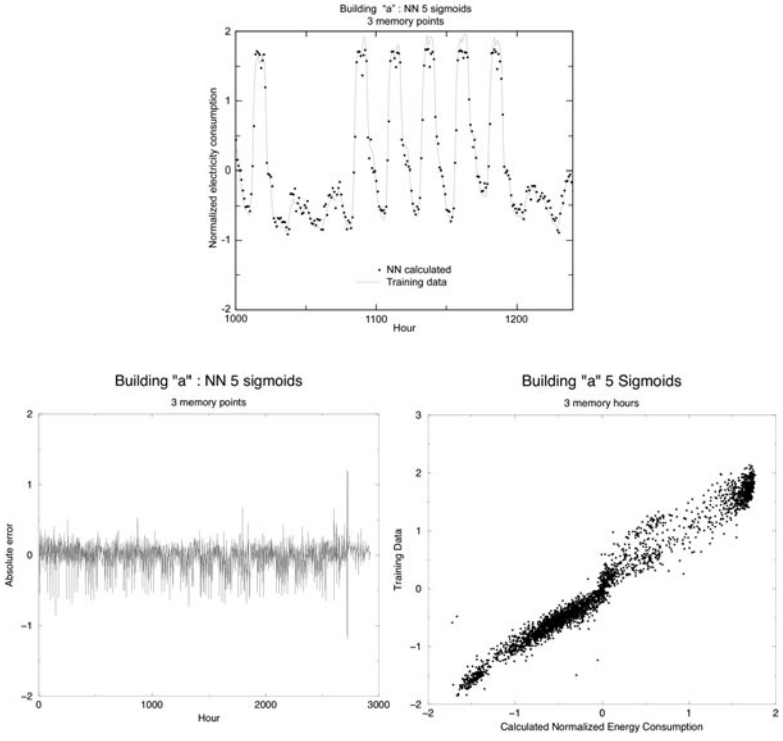

Figure 12.1.3: Results for building "a"; perceptron training with 5 nodes.

differences), and thus we have 10 input variables and 1 output variable. This is then a feedback network. All the variables are normalized to mean 0 and variance 1.

The resulting model, when compared with the normalized training data, has a residual mean error $\sqrt{\Sigma_{i=1}^{m} r_i^2/m} = 0.17$. *Figure 12.1.3 shows some detailed results of the training step, including a comparison of calculated against training output, the absolute error at the training points as a function of the hour counter, and finally a scatter plot of those quantities over the whole range (a perfect fit should give a straight line with a slope of* $45°$*).*

The main observation is that the global fit of this extensive and oscillatory data set with just 5 nodes (i.e., 55 nonlinear and 6 linear parameters) is remarkably good, and the training time per run is still only 24.2 seconds on average on a 800 MHz PC.

12.2 Response surfaces, surrogates or proxies

Now that computer power is such that many complex multi-physics problems (i.e., involving several simultaneous physical phenomena) can be adequately modeled, scientists and engineers are pushing the boundary to the next challenge that requires many simulations, such as those involved in optimal design and material identification. In order to make the solution of these problems practical with current hardware, one has to resort either to large-scale parallelization or, as we describe below, to response surfaces, surrogates or proxies, which are approximations obtained from a limited training set of simulations. Another application of these techniques is the fast evaluation of simulations (i.e., faster than by full high-fidelity modeling), required in many real-time applications, as we exemplify below. Many approximations are used in practice, such as low-order polynomials, but here we will use instead neural networks. Low-order polynomials may not lead to appropriate proxies, and the number of terms grows rapidly with the number of unknowns.

Example 109. *This real data test is taken from a finite element simulation database developed by Weidlinger Associates Inc. to evaluate the progressive collapse potential of reinforced-concrete and steel-frame buildings; see Figure 12.2.1 for an example. The independent variables correspond to configuration parameters (such as bay size, beam type, etc.) and the load applied to the structure. The dependent variable is the response of the structure (e.g., deflection) to such load. Hence, the data set defines the load-response characteristics of the structure for particular configurations, as illustrated in Figure 12.2.2.*

One of the data sets taken from the database and used for illustration herein is given in Table 12.1. The independent variables (x_1, x_2, x_3) *correspond to the beam type, the bay size and the applied load, respectively. The dependent variable,* y_1*, corresponds to the deflection response. Many other response quantities of interest such as rotation, thrust and moment are included in the database, but have been left out of the sample data set for the sake of clarity.*

We have used program VARPRO for training, considering a different number of nodes with sigmoid activation functions in a single hidden layer, fully connected perceptron, with three input and one output node. In Table 12.2 we summarize the results on a 800 MHz Celeron PC, running under the Solaris x86 operating system. For the last two rows, the RMS indicates that we are essentially interpolating the data. This surrogate can then be used to rapidly evaluate the response for other values of the input parameters.

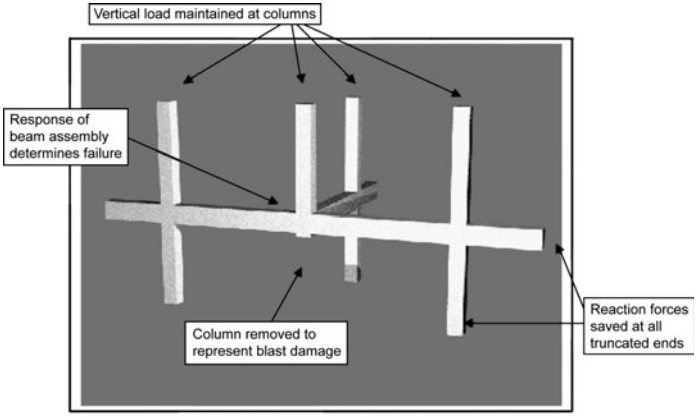
Vertical load maintained at columns
Response of beam assembly determines failure
Column removed to represent blast damage
Reaction forces saved at all truncated ends

Figure 12.2.1: A typical frame unit of a building.

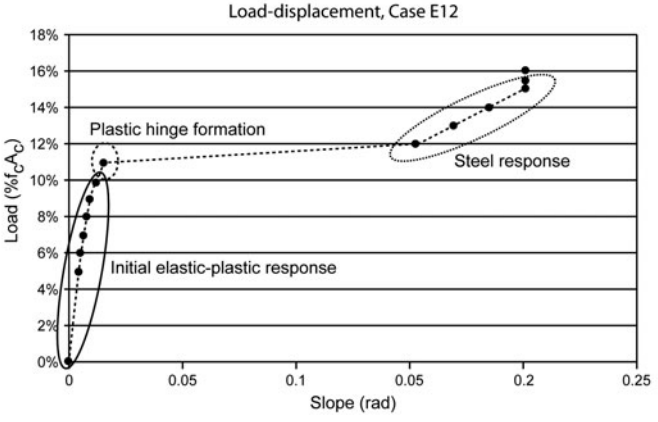

Figure 12.2.2: Load-response characteristics.

x_1	x_2	x_3	y	x_1	x_2	x_3	y
90	20	10	−0.71	90	20	30	−6.50
308	20	80	−0.82	306	20	140	−17.05
824	20	150	−0.58	824	20	400	−20.57
1125	20	200	−0.52	1125	20	550	−19.73
1607	20	300	−0.57	1507	20	800	−24.52
90	25	10	−1.42	90	25	30	−18.54
308	25	20	−0.46	308	25	100	−16.25
824	25	100	−0.67	824	25	300	−23.31
1125	25	150	−0.72	1125	25	400	−20.05
1607	25	250	−0.93	1607	25	600	−25.38
90	30	10	−2.67	90	30	30	−26.61
308	30	20	−0.79	308	30	60	−19.03
824	30	100	−1.30	824	30	250	−30.86
1125	30	150	−1.46	1125	30	300	−17.78
1607	30	200	−1.21	1807	30	450	−22.15
90	20	20	−1.94	90	20	50	−23.59
308	20	100	−3.36	308	20	180	−33.97
824	20	300	−4.32	824	20	500	−41.02
1125	20	400	−3.51	1125	20	685	−40.60
1607	20	600	−4.79	1607	20	928	−41.09
90	25	20	−5.92	90	25	40	−26.62
306	25	40	−0.98	308	25	160	−45.40
824	25	200	−3.38	824	25	400	−49.30
1125	25	300	−4.57	1125	25	550	−49.85
1807	25	450	−5.56	1607	25	710	−53.26
90	30	20	−13.73	90	30	40	−33.77
308	30	40	−1.96	308	30	120	−44.38
824	30	I	−3.60	824	30	350	−59.24
1125	30	250	−7.12	1125	30	500	−63.97
1607	30	350	−6.11	1607	30	685	−67.05

Table 12.1: Data used in example. x_1 = beam type, x_2 = bay size, x_3 = applied load, y = deflection response.

# nodes	RMS	iterations	CPU time in seconds (100 runs)	# nonlinear parameters
2	0.3693	52	2.39	8
4	0.1128	167	18.71	16
6	0.089	214	53.91	24
8	0.0551	262	102.48	32
10	0.0175	300	156.42	40
12	0.0053	287	205.09	48
14	$6.25 \cdot 10^{-13}$	185	251.77	56
16	$1.23 \cdot 10^{-13}$	72	240.4	64

Table 12.2: Results for progressive collapse example. The CPU time is the accumulated time for 100 runs.

12.3 Optimal design of a supersonic aircraft

Optimization problems in many industrial applications are extremely hard to solve in a general manner. Good examples of such problems can be found in the design of aerospace systems. Because of the high level of integration of today's systems and the increase in complexity of analysis and design methods for the evaluation of system performance, such problems are characterized by multi-disciplinary simulations, goal functionals and constraints that are expensive to evaluate, and, in addition, they often have large-dimensional design parameter spaces that further complicate the solution of the problem.

The resulting optimization problems are frequently non-convex, i.e., multi-modal and ill-conditioned, making complete optimizations of complex aircraft configurations prohibitively expensive. Finally, the simulations may require using legacy or commercial codes that have to be used as black boxes that, in particular, may not produce the derivative information required by some optimization techniques.

Multi-modal, ill-conditioned problems require global optimization techniques and regularization [118, 189] and present some of the most challenging problems for robust initialization and ulterior accurate solution. In high-dimensional spaces, the available techniques are problematic at their best and one often must resort to surrogate models, divide and conquer techniques and parallel computing, in order to even have a chance to solve the problem in a reasonable time [2, 45, 188, 189].

As we indicated above, the most common surrogate models consist of low-degree polynomial approximations. These are inadequate as surrogates for highly discontinuous functions such as supersonic boom. In this section we use neural networks to produce surrogates that more faithfully and successfully reproduce such functions. As before, we also use a fast training

tool for generating the NN via the variable projection method.

Example 110. *We consider the generation of sample data for the aerodynamic and boom characteristics of a generic supersonic aircraft configuration, the fitting of this data using neural networks, and the use of the resulting surrogate models in a representative design optimization problem. For this test example, some of the most relevant design parameters were selected, as described below. Although we use only a small number of design variables relative to a realistic aerospace vehicle design, the problem has all of the elements necessary to exercise and evaluate the proposed initialization and optimization techniques and also to appreciate the performance of the surrogates.*

The two goals in this bi-objective problem are to maximize the total range of the aircraft and to minimize the perceived loudness of the ground boom signature (measured in dBA), while satisfying a number of mission and buildability constraints, such as

- *Structural integrity of the aircraft for a $N = 2.5$ g pull-up maneuver.*
- *Takeoff field length $<$ 6,000 ft.*
- *Landing field length $<$ 6,000 ft.*

For high-fidelity aerodynamic modeling we use the A502 solver, also known as PanAir [38, 39], a flow solver developed at Boeing to compute the aerodynamic properties of arbitrary aircraft configurations flying at either subsonic or supersonic speeds. This code uses a higher-order (quadratic doublet, linear source) panel method, based on the solution of the linearized potential flow boundary-value problem. Results are generally valid for cases that satisfy the assumptions of linearized potential flow theory – small disturbance, not transonic, irrotational flow, and negligible viscous effects. Once the solution is found for the aerodynamic properties on the surface of the aircraft, A502 can then easily calculate the flow properties at any location in the flow field, hence obtaining the near-field pressure signature needed for sonic boom prediction. In keeping with the axisymmetric assumption of sonic boom theory, the near-field pressure can be obtained at arbitrary distances below the aircraft [5].

The high-fidelity method for computing the ground boom signature is shown in Figure 12.3.2. At the near-field plane location, the pressure signature created by the aircraft is extracted and propagated down to the ground using extrapolation methods based on geometric acoustics.

The location of the near-field must be far enough from the aircraft, so that its flow field is nearly axisymmetric and there are no remaining diffraction effects, which cannot be handled by the extrapolation scheme. Since A502/Panair only uses a surface mesh for all of its calculations, it is able

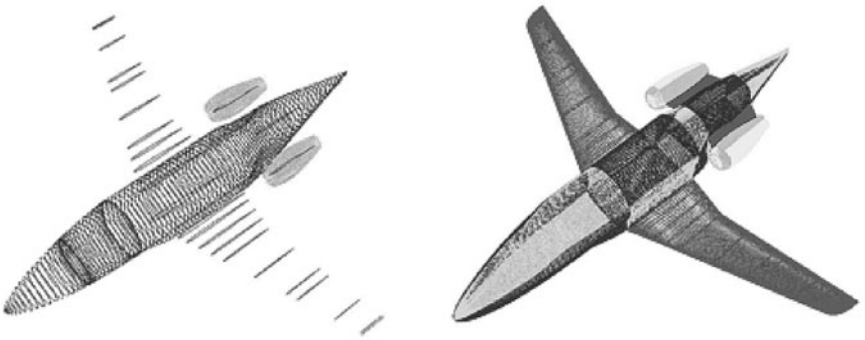

Figure 12.3.1: Un-intersected components of a transonic business jet configuration (left) and intersected components forming the outer mold line, a well-defined three-dimensional surface that provides the outer geometry of the plane (right).

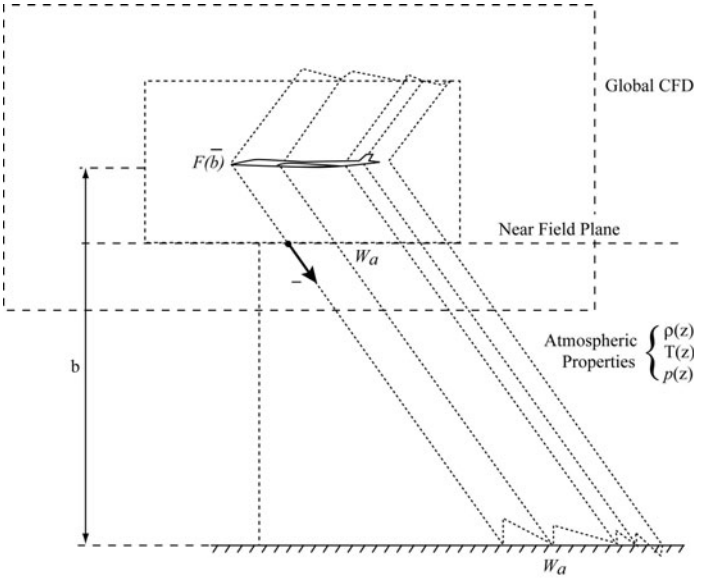

Figure 12.3.2: Sonic boom propagation procedure.

to obtain near-field pressures at arbitrary distances without changes in the computational cost. In this work we are using the Sboom [19] extrapolation method to propagate near-field signatures into ground booms. The sonic boom extrapolation method accounts for vertical gradients of atmospheric properties and for stratified winds (although the winds have been set to zero in this example). The method relies on results from geometric acoustics for the evolution of the wave amplitude and utilizes isentropic wave theory to account for nonlinear waveform distortion due to atmospheric density gradients and stratified winds.

Past research on low-boom aircraft design focused on reducing the magnitude of only the initial peak of the ground boom signature [1, 46]. This requirement, which had been suggested as the goal of the DARPA-sponsored Quiet Supersonic Platform (QSP) program ($\Delta p_0 < 0.3\, pound \times foot^2$), neglects the importance of the full signature, which depends on the more geometrically complex aft portion of the aircraft, where empennage, engine nacelles and diverters create more complicated flow patterns. Moreover, such designs often have two shock waves very closely following each other in the front portion of the signature [39, 47], a behavior that is not robust and is therefore undesirable. For these reasons, we are computing the perceived loudness (measured in dBA) of the complete signature. Frequency weighting methods are used to account for the fact that humans do not have an equal response to sounds of different frequencies. In these calculations, less weight is given to the frequencies to which the ear is less sensitive. In addition, all signatures computed are post-processed, to add a physical rise time across the shock waves yielding loudness numbers that are more representative of those perceived in reality.

Generation of response surfaces (surrogates)

Using the tools described above, 450 configurations obtained via Latin hypercube sampling (LHS) were generated at a high computing cost. This sample input-output data set was fitted using a neural network (NN) in order to produce surrogates that could be used for optimization in a more economical form. The NN was a single hidden layer perceptron with sigmoid activation functions that provided a general nonlinear model. We used for its fast training (i.e., determination of the NN parameters) the variable projection algorithm (VARPRO) to solve the resulting nonlinear least squares separable problem in order to generate a reduced cost approximation of the objective space. This was combined with a global optimization algorithm, since the resulting problems are generally multimodal [37].

The training of a single hidden layer neural network using sigmoid activation functions leads to a separable nonlinear least squares problem, in which the unknown parameters in the network are determined as the best fit of the training data in the l_2 sense (i.e., the input/output data contained

in the simulation database). This results in a surrogate function that is a linear combination of sigmoids:

$$S(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2, \ldots) = \sum_j a_j \frac{1}{1 + e^{-(\boldsymbol{x}^T \boldsymbol{\alpha}^j + \alpha_0^j)}},$$

where $\boldsymbol{x} \in \mathbb{R}^k$ corresponds to the input or design parameters, a^j are the linear parameters and $\boldsymbol{\alpha}^j \in \mathbb{R}^k$ are the nonlinear parameters. The data used corresponds to a representative supersonic business jet configuration. For each output, the training files contain 300 data points, while a test set of an additional 150 data points is used for evaluation of the fit. There are eight independent or input variables, namely:

- *Wing reference area;*
- *Wing aspect ratio;*
- *Longitudinal position of the wing;*
- *Wing sweep angle;*
- *Lift coefficient at initial cruise;*
- *Lift coefficient at final cruise;*
- *Altitude at initial cruise; and*
- *Altitude at final cruise.*

There are 10 dependent variables that we want to approximate with neural networks, namely:

- *Drag coefficient at initial cruise;*
- *Sonic boom initial rise at initial cruise;*
- *Sonic boom sound level at initial cruise without rise time modification;*
- *Sonic boom sound level at initial cruise with first type rise time modification;*
- *Sonic boom sound level at initial cruise with third type rise time modification;*
- *Drag coefficient at final cruise;*
- *Sonic boom initial rise at final cruise;*
- *Sonic boom sound level at final cruise without rise time modification;*

Output	Max. value	#Sig-moids	RMS training	RMS test	Time (sec)	Max. Res.
1	–	8	0.0000865	0.00033	872	0.0009
2	–	8	0.0033	0.206	831	0.15
3	–	10	0.406	1.2	1872	1.5
4	95	10	0.848	2.0	2332	3.0
5	95	10	0.541	2.01	2393	1.9
6	0.017	10	0.0000721	0.000229	2782	0.00023
7	3	8	0.0364	2.66	1393	0.26
8	99	10	0.414	1.14	1773	1.9
9	94	10	0.747	4.46	2090	3.2
10	94	10	0.545	1.71	1917	2.2

Table 12.3: Neural Network results for surrogate functionals.

- *Sonic boom sound level at final cruise with first type rise time modification; and*
- *Sonic boom sound level at final cruise with third type rise time modification.*

We scale the input variables so that they have zero mean and variance equal to 1. In Table 12.3 we show the results of training and testing for each one of the 10 outputs.

Using these surrogates we conduct a number of multiobjective genetic algorithm optimizations that attempt to simultaneously maximize the range (in nautical miles) and minimize the perceived noise level at the ground (in dB), while satisfying all the constraints. The NSGA-II evolutionary algorithm [59] is used to generate an approximation to the Pareto front, and a summary of the most promising results can be seen in Figure 12.3.3.

The initial population is composed of 64 alternative designs that are randomly distributed within the design space. The level of performance of these designs is represented by the open circles in Figure 12.3.3. These circles constitute a pair of (range, loudness) for each of the aircraft in question. Note that the optimization algorithm attempts to drive all designs toward the upper left-hand corner of the graph. Of the 64 initial design alternatives only some represent feasible designs, in the sense that they satisfy the constraints of the problem (takeoff and landing field lengths $<$ 6,000 ft). The open circles represent designs that do not meet the design constraints and are therefore infeasible and must be discarded. That is the reason why some open circles appear to have extraordinary performance. Notice, again, that the baseline configuration does not meet the constraints

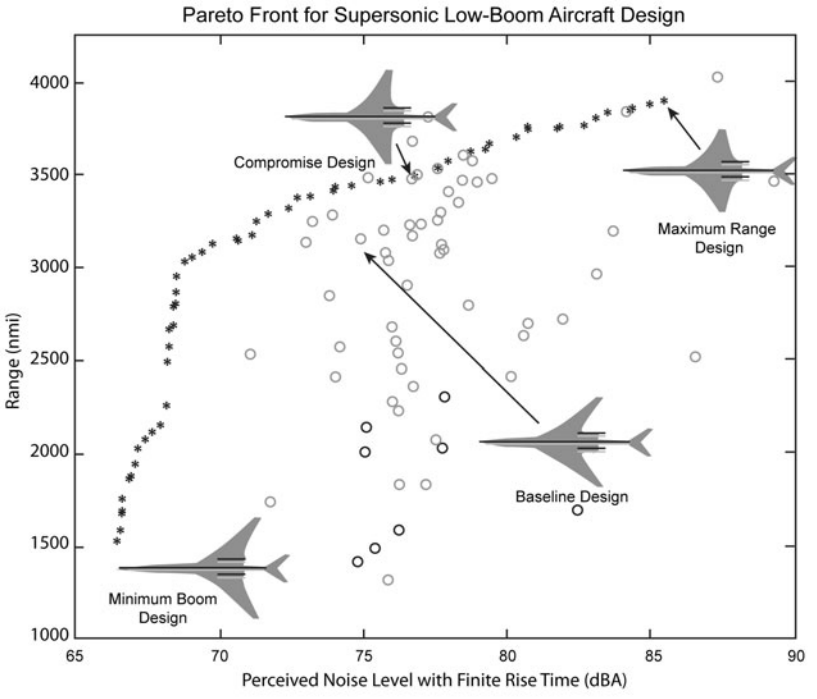

Figure 12.3.3: Results in objective space for the multiobjective genetic algorithm optimization. Initial population: circles. Final population: asterisks (all feasible designs).

(takeoff field length in this case) of the optimization problem.

In the genetic algorithm we choose a crossover probability for the variables of 0.75 and a mutation probability of 0.125. The distribution indices for both the crossover and mutation operations are taken to be 12% and 25%, respectively. The population is allowed to evolve over 1,000 generations. As the generations are evolved, the entire population migrates toward feasibility and attempts to maximize their range and minimize their noise level simultaneously. The result is the Pareto front of solutions given by the asterisks in the graph. This Pareto front represents the set of non-dominated solutions of the problem (no other solution is better than any of those solutions in both performance measures simultaneously). Notice that all asterisks on the Pareto front correspond to aircraft that satisfy the constraints of the problem (that was not the case for the initial population).

The designs/aircrafts on the Pareto front provide the designer with alternatives to trade-off the relative benefits of each of the two (conflicting) objective functions: one can choose an aircraft with very low boom signature, but also very low range, or an aircraft with extremely high range, but rather poor sonic signature, or something in between. Note that the shape of the Pareto front is quite unique: it is initially very steep and then its slope appears to decrease to a smaller value (the shape is reversed from the usual L-curve because we are minimizing one objective and maximizing the other).

In fact the Pareto front is well approximated by two straight-line segments with a knee located at a loudness level of approximately 67.5 dBA. The interpretation is clear: it pays off to sacrifice a little bit of the quietness of the aircraft to attain significant improvements in its range. After the loudness increases beyond 67.5 dBA, modest improvements in range are obtained at the expense of very large increases in sonic boom loudness.

Observe that this optimization task would be impossible with current computer machinery if we would need to use the high-fidelity simulations for all the 64,000 evaluations involved in the calculation. Amazingly, the optimal design chosen, when verified with the high-fidelity simulators, had a performance within 2% of it, so it would not really be necessary, for practical engineering purposes, to perform any additional refinements.

12.4 NMR spectroscopy

Nuclear magnetic resonance (NMR) in liquids and solids was discovered more than 50 years ago by Ed Purcell (Harvard) and Felix Bloch (Stanford), who shared the Nobel Prize in Physics in 1952 for their work. NMR is now a fundamental analytical tool in synthetic chemistry, plays an important role in biomedical research, and has revolutionized modern radiology and neurology. Its applications outside the laboratory or medical clinic are as

diverse as imaging oil-wells, analyzing food, and detecting explosives.

A typical NMR experiment involves placing the sample under study in a strong magnetic field, which forces the magnetic moments or spins of all the nuclei in the sample to line up along the main applied field and precess around this direction. The spins precess at the same frequency but with random phases. Pulses of radiofrequency (RF) magnetic fields are then applied that disturb the spin alignments but make the phases coherent and detectable. As this state precesses in the magnetic field, the spins emit radio-frequency radiation that can be analyzed to reveal the structural, chemical and dynamical properties of the sample. The idea of applying strong RF pulses is due to H. C. Torrey and E. L. Hahn. Higher and higher magnetic fields, of the order of teslas, have been used in order to increase the sensitivity of the method until just recently, when researchers in the United States and Germany have shown that MR can be performed with fields in the microtesla range, by pre-polarizing the nuclei and using a superconducting quantum interference device (SQUID) to detect individual flux quanta. In a magnetic field of 1.8 microtesla, the researchers observe proton magnetic resonance in a liquid sample at about 100 Hz with an astonishing degree of sensitivity.

In vivo NMR spectroscopy has the strongest connection and owes most to the variable projection methodology. In 1988, van der Veen et al. [244], a group of researchers at Delft University and Phillips Medical Systems in the Netherlands, published a very influential paper on the accurate quantification of in vivo NMRS data, using the variable projection method and prior knowledge (constraints). According to Google Scholar, as of August 2012 this paper had more than 335 citations.

The problem here is to fit NMR spectra in the time domain, using models whose parameters have physical significance. The most commonly used model in NMRS is a linear combination of exponentially damped sinusoids, but other types of nonlinear functions are also considered:

$$y_n = \sum_{k=1}^{K} a_k e^{j\phi_k} e^{(-d_k + j2\pi f_k)t_n} + e_n, \quad n = 0, 1, \ldots, N,$$

where $j = \sqrt{-1}$, a_k is the amplitude, ϕ_k is the phase, d_k is the damping factor, and f_k is the frequency.

Van der Veen et al. consider NMRS measurements of human calf muscle and human brain tissue and compare the FFTs of the original data with a linear prediction and SVD decomposition, and VARPRO with and without prior knowledge. The best results are obtained with VARPRO plus prior knowledge, which was difficult or impossible to impose on previous simpler approaches. They also list as desirable features the fact that starting values of the amplitudes of the spectral components are not required and that there are no restrictions on the form of the model functions.

The model, without prior knowledge, involves 256 complex data points and 11 exponentials, for a total of 44 parameters divided equally between linear and nonlinear. In order to obtain good initial values, the time signal was Fourier transformed and displayed. Then, an interactive peak-picking was performed. This provided good initial values for the frequency and the damping of each peak. When the full functional is minimized, one also needs initial values for the linear parameters, and these are obtained by solving the LSQ problem obtained by evaluating the nonlinear part at the chosen initial values, a very wise choise as indicated in the original paper [101], since this is much better than using arbitrary values.

Van der Veen et al. first consider three different optimization methods for the full and the reduced problem, using variable projection with the Kaufman improvement:

- The original VARPRO algorithm with the Levenberg-Marquardt implementation,
- a secant type code NL2SOL [65],
- and LMDER, a modern Levenberg-Marquardt implementation from MINPACK [263].

According to their results, NL2SOL with the separated functional seems to be the most reliable, even for large levels of noise. MINPACK's routine is almost as reliable and systematically faster. We should say here that the reported average times for this problem are about one minute on a SUN ULTRA2 (200 MHz), so in current and future platforms the differences in performance are negligible.

The most important conclusion drawn there is that if a VARPRO-type code is to be used, it should include the Kaufman simplification and should take advantage of the advances in numerical optimization. The MINPACK NLLS solver LMDER or the Gay-Kaufman implementation are good candidates for replacement codes. See, however, the recent results of [172].

Some years later, a group in Leuven, Belgium [245, 246, 247], made a comparative study for this problem, including artificial noise and also using prior knowledge. They examined one of the data sets considered earlier by van der Veen et al. They added different levels of white Gaussian noise, small (5%), medium (15%) and large (25%), and considered 300 runs per level in a Monte Carlo simulation.

The study of Vanhamme et al. considers also three different methods to obtain starting values: HSVD, a fully automatic parameter estimation method that combines a state-space approach with SVDs: pick1, the peak-picking method described above, and pick2, a more careful (and expensive) version of pick1. Since the influence of these different initial value choices seems to be method-independent, only results for MINPACK were

presented. The conclusions were that the desirable automatic procedure works very well for low and medium noise levels, but not as well for high levels of noise. The procedures pick1 and pick2 are similar in performance and reliability, with a slight edge for pick2.

Finally, they considered the effect of using prior knowledge about the problem. In prior1, the number of linear variables is reduced to 11 by noting that all the peaks have a phase of 135°. In prior2 all the known prior knowledge is used to eliminate variables, obtaining a problem with 5 linear and 12 nonlinear parameters. Results are given for MINPACK only and here, using prior1 variable projection has a slight edge in performance for small and medium noise levels, although for medium and high levels of noise there is a deterioration in the reliability. For prior2, variable projection is consistently more reliable than the full functional approach, with a slight penalty in performance (under 8%), which has now come down to under 10 seconds of CPU time.

As a conclusion to this study the authors suggested using the full functional instead of the reduced one, and they proceeded to write their own solver AMARES to do that. This new solver includes some other features special to the problem that were not present in [244]. The conclusions of Vanhamme et al. indicate that with the Gay and Kaufman implementation of variable projection, the balance clearly tilts in favor of the reduced functional, both in speed and reliability. Both VARPRO and AMARES are currently offered in the MRUI system, which according to the development group, "more than 780 research groups worldwide in 53 countries benefit from."

12.5 Piezoelectric crystal identification

The physical model

A piezoelectric transducer converts electrical signals into mechanical deformations to generate sound waves, or conversely, it converts mechanical deformations into electric signals in order to monitor and record incoming sound waves. Piezoelectric transducer arrays are used as ultrasound sources and receivers for medical imaging, sonar and oil exploration applications. There is extensive research in the ultrasound industry to validate and design new prototype piezoelectric transducers with optimal performance characteristics.

Example 111. *A homogeneous piezoelectric crystal model is defined by ten elastic, electromagnetic and coupling parameters, by the geometry of the sample and that of the electrodes. One can use a finite element (FE) time domain code for the numerical modeling of the device response. The numerical model can be used to provide insight into the performance of a*

specific design, but also, via a nonlinear inversion process, it can be used to determine the material property parameters of an unknown crystal from complex impedance measurements. These properties are essential for forward simulation in design processes.

The problem that we will consider involves a nonlinear least squares inversion algorithm to improve on the accuracy of some nominal initial values of the parameters of a given transducer crystal, using measured values at successive time intervals of the resulting impedance, when a small voltage is applied to the sample.

We accomplish this task by coupling forward modeling and optimization tools, namely, PZFLEX [205], a coupled finite element time domain code for the elastic and electromagnetic parts of the problem, and PRAXIS, a general unconstrained minimization code for the nonlinear least squares fit [31]. No explicit derivatives of the goal functional are required by this code. Bound constraints are imposed in order to limit the variability of the parameters to physically meaningful values, and, since PRAXIS is an unconstrained optimization code, these constraints are introduced via a change of independent variables, so that when the new variables run over the whole space (i.e., unconstrained), the physical parameters stay in the desired box. This can be achieved, for example, by using the transformation:

$$\alpha = \underline{\alpha}/(1 + e^{x}) + \bar{\alpha}/(1 + e^{-x}),$$

which has the desired properties, i.e., when $-\infty < x < \infty$, then $\underline{\alpha} < \alpha < \bar{\alpha}$. The inverse relationship is

$$x = -ln[(\bar{\alpha} - \alpha)/(\alpha - \underline{\alpha})],$$

and that is the change of variables that we use to convert the box constrained problem into an unconstrained one.

In order to characterize the material properties of a given crystal, current practice uses several IEEE standard shapes (corresponding essentially to asymptotic limit cases) to determine different groups of parameters at a time by trial and error. These samples are fairly expensive and sometimes hard to manufacture. Given the analysis tools at our disposal, we will use shapes different from the IEEE ones in order to see whether we can determine as many parameters as possible at once and in an automated manner. For this purpose and as a pre-process, we propose to carry on sensitivity analyses of different crystal shapes, to find configurations for which as many as possible of the parameters can be determined at once.

For this analysis we consider the linearized model, which is represented by the Jacobian of the Fourier transform of the impedance values with respect to model parameters. These derivatives are estimated by finite differences. Since the number of parameters is small, we can apply a direct singular value decomposition (SVD) based solver to this rectangular matrix,

calculated at a reference value of the parameters (in practice all these parameters are known within 10% of their true values, for a given material). As explained in Section 9.7 (see also [36, 187, 192, 200]), we can use the SVD to rank the relative relevancy of each parameter, for a given data set. This can be used as a guide to design geometrical configurations and placement of electrodes, so that the measured impedance is sensitive to as many parameters as possible, thus minimizing the number of samples that need to be built in order to determine material properties accurately.

PZFLEX modeling

Operational emphasis for imaging transducers is broadband (impulsive) rather than narrowband (continuous wave). Transducers are currently available for diagnostic imaging and Doppler velocity measurement, as well as for a host of specialty applications (intracavity, biopsy, etc.) and disease treatment (lithotripsy, hyperthermia, tissue ablation). Over the past two decades the ultrasound industry has done a remarkable job in developing and refining these devices, using a combination of semi-analytical design procedures and prototype experiments.

However, it is apparent to many that conventional design methods are approaching practical limits of effectiveness. The industry has been slowly recognizing discrete numerical modeling on the computer as a complementary and useful tool, i.e., using virtual prototyping instead of, or complementing, actual laboratory experimentation. Today, nearly all of the major ultrasound system companies are experimenting with finite element models using commercial packages like ANSYS, or by writing their own codes. Most have enjoyed only limited success at significant development and/or simulation costs.

We suggest that the main source of difficulty is universal reliance on classical implicit algorithms for frequency-domain and time-domain analysis based on related experience with shock and wave propagation problems. In general, implicit algorithms are best suited to linear static problems, steady-state vibrations and low-frequency dynamics. A much better choice for transient phenomena, linear or nonlinear, is an explicit time-domain algorithm, which exploits the hyperbolic (wave) nature of the governing differential equations.

The finite element method reduces the electromechanical partial differential equations (PDEs) over the model domain to a system of ordinary differential equations (ODEs) in time. This is done by using one of the nearly equivalent integral formalisms: virtual work, weak form, Galerkin's method, weighted residuals, or less formally, using pointwise enforcement of the conservation and balance laws. The result is that spatial derivatives in the PDEs are reduced to a summation of "elemental" systems of linear algebraic equations on the unknown field values at nodes of the finite element

discretization.

The continuum elements used in the PZFLEX code are 4-node quadrilaterals in two-dimensions and 8-node hexahedrons in three-dimensions. The unknown field over an element is represented by low-order shape functions determined by nodal (corner) values, i.e., bilinear in two-diensions and trilinear in three-dimensions. Using a minimum of 15 elements per wavelength limits wave dispersion errors to less than 1%*. Experience has shown that these choices offer the most robust basis for large-scale wave propagation analysis in structural and isotropic or anisotropic continuum models.*

When transient signals are of principal interest, the most direct solution method is step-by-step integration in time. There are many ways to evaluate the current solution from known results at previous time steps. Implicit methods couple the current and previous time step solution vectors and hence, a global system of equations must be solved at each time step. Their advantage is unconditional stability with respect to time step. By contrast, explicit methods decouple the current solution vectors, eliminating the need for a global system solve, but they are only conditionally stable, i.e., there is a time step restriction (the Courant-Fiedrichs-Levy (CFL) condition) above which the method is unstable.

The caveat for integration of wave phenomena is that solution accuracy requires a time step smaller than one-tenth the period of the highest frequency to be resolved. This is close to the CFL stability limit for explicit methods and effectively removes the principal advantage of implicit integration. Explicit integration of the field equations involves diagonalizing the uncoupled mass and damping matrices, using nodal lumping, replacing the time derivatives with finite differences and integrating using a central difference scheme (second-order accurate). For stability the time step must be smaller than the shortest wave transit time across any element (CFL condition). More details on the finite element approach can be found in [36].

The nonlinear least squares problem

Given a homogeneous piezoelectric crystal, a small voltage is applied and the resulting impedance is calculated from measurements as a digitized function of time. This function is fast Fourier transformed and the resulting complex samples constitute the observed data, $\{I_i^o\}$, $i = 1, \ldots, m$. *Given a vector of model parameters* $\boldsymbol{\alpha}$ *and using the finite element simulator described above, we can calculate a similar response that we call* $\{I_i^c(\boldsymbol{\alpha})\}$.

In the present problem, we will improve the accuracy of some of the parameters that define a particular transducer, by applying a nonlinear least squares inversion to this measured data, namely:

$$\min_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}) = \min_{\boldsymbol{\alpha}} \sum_{i=1}^{m} (I_i^o - I_i^c(\boldsymbol{\alpha}))^2.$$

Not all the $\boldsymbol{\alpha}$'s are physically feasible; there is only a range $[\underline{\boldsymbol{\alpha}}, \overline{\boldsymbol{\alpha}}]$ of parameters that corresponds to possible materials and the minimization includes, therefore, bound constraints:

$$\min_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}), \qquad \alpha_i \in [\underline{\alpha}_i, \overline{\alpha}_i], \quad i = 1, \ldots, n.$$

Sensitivity analysis

As the manufacturing costs of transducer prototypes is high, we use a linearized sensitivity analysis, as described in Section 9.7, to determine a priori which parameters can be calculated from measured impedances on several crystal samples with different geometries. A summary of the procedure follows: Given a parameter vector $\boldsymbol{\alpha}$, calculate $J(\boldsymbol{\alpha})$ by finite differences and its SVD.

1. *Let the matrix of the right singular vectors, scaled by the corresponding singular values, be $\overline{V} = V\Sigma$.*

2. *Inspect the rows of $\overline{V}$ and select the elements above a certain threshold. Choose the indices of the variables in parameter space corresponding to these entries to form the subset $\boldsymbol{\alpha}$ of parameters that are most influenced by the data set. Observe that this does not establish a direct correspondence between small singular values and parameters, but rather one between singular values and linear combinations of parameters, given by the rotated variables.*

Numerical results

We consider a cylinder of piezoceramic material. The finite element modeling assumes two planes of symmetry: axial at half height and radial, in order to optimize the computation, which then becomes two-dimensional. For a fixed diameter of 10 mm we consider several heights, giving aspect ratios (diameter/height) between 20/1 (corresponding to one of the IEEE standard shapes) and 1/2. By performing the analysis described in the previous section we have calculated the results shown in Table 12.4, where the numbers under the various parameters indicate how well determined they are when the real and imaginary part of the impedance are used as the data set. A value larger than 0.1 indicates a well-determined parameter, while a value less than 0.01 indicates a poorly determined one.

Asp. rat.	ep11	ep33	s11	s12	s13
20/1	6e-4	**0.62**	**0.79**	**0.2**	0.04
5/1	0.07	**0.3**	**0.89**	**0.25**	0.07
1/1	0.05	**0.5**	**0.4**	**0.12**	**0.3**
1/2	0.046	**0.42**	**0.4**	**0.12**	**0.21**
Asp. rat.	s33	s44	d15	d13	d33
20/1	0.07	8e-4	3e-6	**0.51**	0.07
5/1	0.07	0.01	0.01	**0.58**	0.04
1/1	**0.44**	0.075	0.087	**0.41**	**0.66**
1/2	**0.56**	0.086	0.083	**0.26**	**0.74**

Table 12.4: Sensitivity analysis for a cylinder of piezoceramic material. The bold numbers indicate well-determined parameters.

Param	Name	Target	Final	% error
ep11	dielectric 1	$0.213 \cdot 10^{-7}$	$0.213 \cdot 10^{-7}$	0.12
ep33	dielectric 2	$0.297 \cdot 10^{-7}$	$0.296 \cdot 10^{-7}$	-0.33
s11	compliance 1	$1.559 \cdot 10^{-11}$	$1.530 \cdot 10^{-11}$	2.11
s12	compliance 2	$-0.441 \cdot 10^{-11}$	$-0.491 \cdot 10^{-11}$	12.2
s13	compliance 3	$-0.819 \cdot 10^{-11}$	$-0.758 \cdot 10^{-11}$	6.81
s33	compliance 4	$2.0 \cdot 10^{-11}$	$2.01 \cdot 10^{-11}$	0.68
s44	compliance 5	$4.48 \cdot 10^{-11}$	$4.49 \cdot 10^{-11}$	0.3
d15	piezoe. stress 1.	$7.19 \cdot 10^{-10}$	$7.2 \cdot 10^{-10}$	0.32
d13	piezoe. stress 2.	$-2.895 \cdot 10^{-10}$	$-2.63 \cdot 10^{-10}$	-9.08
d33	piezoe. stress 3.	$6.047 \cdot 10^{-10}$	$6.521 \cdot 10^{-10}$	7.79

Table 12.5: Target and final parameters.

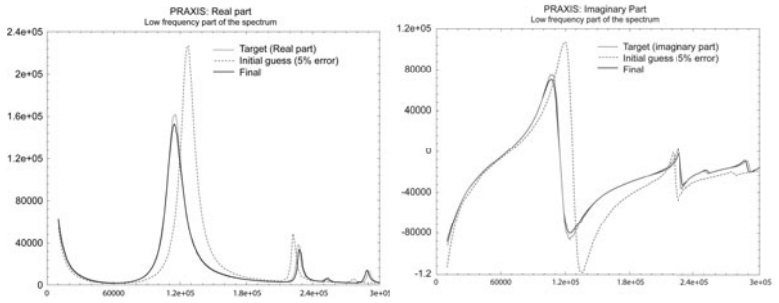

Figure 12.5.1: Initial, target and final, real and imaginary parts of the impedance spectrum for low frequencies.

These results indicate that, considering truly three-dimensional shapes, such as those with 1/1 or 1/2 aspect ratios, provides a better way to estimate more of the relevant parameters at once than using the IEEE shapes. We consider now the disk with aspect ratio $1/2$ *to test our parameter determination code. We will use PRAXIS to least squares fit a data set consisting of the real and imaginary parts of the FFT of the impedance, which in the range* 1 *KHz to* 1 *MHz is represented in digital form by* 1341 *unequally spaced samples. One important feature of PRAXIS is that it does not require derivatives with respect to the unknown parameters, which are not available in the large-scale code PZFLEX.*

Because of symmetries and the fact that the material is considered homogeneous, the problem can be cast as a two-dimensional problem, and for the wavelengths, materials and model size involved, a mesh of 25×50 *elements is appropriate. For a driving frequency of* 3.75 *MHz, the elastic part of PZFLEX requires* 7091 *time steps. On a* 300 *MHz Pentium II computer under the SOLARIS operating system an average solve takes approximately 75 seconds.*

We first generate synthetic data by running PZFLEX with the set of parameters shown in Table 12.5 (the "target values"). Then we perturb these parameter values by 5% and call them the "initial values." The objective is to see how many of the target values of these parameters we can recover, say, to 1% accuracy or better. We also use a change of variables strategy to force the parameters to stay within a $\pm 8\%$ *box around the initial guess. After 172 evaluations (i.e., PZFLEX solves; a 3.5 hours job), the results are shown in Table 12.5. The initial and final residual mean squares were* 64844 *and* 1068*, respectively.*

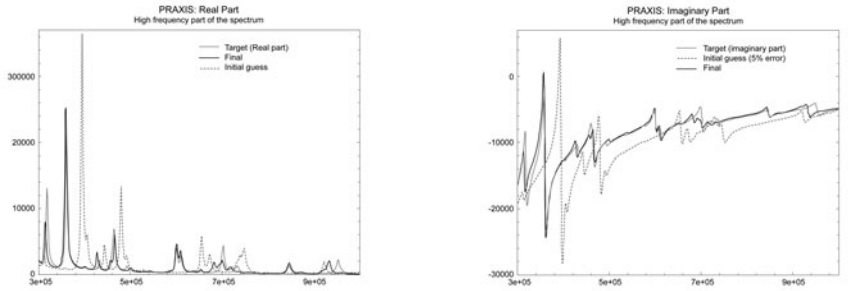

Figure 12.5.2: Initial, target and final, real and imaginary parts of the impedance spectrum for high frequencies.

In Figures 12.5.1 and 12.5.2 we show the plots of the target and final, real and imaginary parts of the impedance spectrum for low and high frequencies separately. Observe that there is a change of scale going from low to high frequency in order to enhance the details. In this typical example we have been able to recover half of the parameters to the desired accuracy, and an additional parameter has been improved by more than 50%, while the estimates for the remaining parameters have worsened. Observe that our bound constraints allow for a maximum error of 16% in the worst-case scenario.

12.6 Travel time inversion of seismic data

A seismic survey for the exploration of hydrocarbons consists of generating elastic waves and recording their back-scattering from the material discontinuities in the underground, with the purpose of obtaining information about their composition and geological structure. By processing and interpreting the obtained data, it is sometimes possible to extract information of the travel time of different coherent events (reflections), which in turn can be used to improve the knowledge of the material properties of the subsoil. Only a restricted number of measurements is available, and these problems are therefore highly underdetermined. In principle, the process will estimate the rock parameters as well as the subsurface structure only at a discrete set of points or as a finite-dimensional parametrized analytic model.

The objective of the travel time inversion process is to improve on the

accuracy of the parameters $\boldsymbol{\alpha}$ that appear in the description of reflector positions and inhomogeneous velocities within a given geological region using seismic data. For this task we require a complete set of forward modeling tools that include interactive structural model construction and three-dimensional seismic ray tracing (see [189]).

Given a set of parameters that describe the geological medium, we can compute via seismic ray tracing the approximate time response to a source excitation, as measured at a geophone receiver array. We aim to determine, from a set of m observed and interpreted arrival times t_i^0 measured at the receivers, improved values of the n model parameters $\boldsymbol{\alpha}$, such that the computed arrival times $t_i^c(\boldsymbol{\alpha})$ match the observed times "as well as possible" in, for example, the l_2-norm sense. The resulting nonlinear least squares problem is

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{m} (t_i^0 - t_i^c(\boldsymbol{\alpha}))^2.$$

Travel times between sources and receivers and derivatives with respect to the parameters are required. Calculation of travel time derivatives can be accomplished in this case economically with information produced during the ray tracing (see [189]). The vector of parameters $\boldsymbol{\alpha}$ can be subdivided into variables that define the interfaces and variables that describe material properties in the regions between the interfaces. A seismic survey usually extends over a considerable area, where many shots are placed together with local receiver arrays, leading to small aperture beams of rays that only "see" a limited part of the model at a time.

If the model is parametrized in a local way, for instance, using B-splines, then this leads to a natural setup for dealing with the resulting very large-scale problems (millions of observations, thousands of parameters), using the block partitioning approach described in an earlier chapter. If we partition the data set by shots, we can then determine the parameters that are sensed by each subset of data, thus obtaining as many subproblems of the same kind as the global one as there are shots. A block nonlinear asynchronous Gauss-Seidel iteration leads to a solution of the global problem that is naturally parallelizable in a network of computers (see [187]).

Example 112. Application to synthetic data

We consider a synthetic data set corresponding to a geothermal field in Indonesia. The aim of the exercise is to demonstrate the viability of the travel time inversion to provide velocity sections in regions with complex near-surface geology and significant topographic relief, using both turning-ray and reflection data. The data are based on an acquisition geometry and geological structure for the Silangkitang, Indonesia, field area. By synthetic data we mean that, given a structural model and a set of model parameters $\boldsymbol{\alpha}^{\text{target}}$ *for the velocity, we generate by ray tracing a set of turning-ray and*

reflected travel times. We call those travel times "the observed data." We then alter the parameters, generating an initial velocity and improve it step by step using the methods described above. This is a standard procedure to generate problems with known solution in order to test and validate a complex algorithm.

The Silangkitang two-dimensional section has a lateral extent of 4000m, and we invert it for a depth of 1000m. The shot gather data are given by as many records as data points, in the format

$$x_{\text{source}},\ y_{source},\ z_{\text{source}},\ x_{\text{receiver}},\ y_{\text{receiver}},\ z_{\text{receiver}},\ \text{travel time}.$$

From these coordinates we deduce the topography and fit a B-spline curve to represent it. In this model there was no reflection data available, but we put an artificial reflector at 1000 m depth to create the synthetic data. We use 50 shots, distributed along the surface of the model, and 160 fixed receivers spaced at 25m.

As we indicated above, we start with the turning-ray data (also referred to as refracted arrivals or diving rays), which join source to receivers without reflecting. In order to create a one-dimensional (variation in depth only) initial velocity model, we examined the travel time curves for a few shots and decided on an appropriate gradient:

$$v^{\text{init}}(z) = v_0 + gz.$$

We add to this background gradient a one-dimensional B-spline correction with 8 equally spaced basis functions. With this gradient and the correction set to zero, we proceed to perform one sweep of a block Jacobi inversion procedure, i.e., for each block we solve the nonlinear least squares problem and save the updated parameters. Once all the blocks are processed we average those parameters that had more than one correction. Block Jacobi is a synchronous procedure. Currently, we favor a Gauss-Seidel approach that does averaging on the fly for parameters that appear in more than one block.

The next step consists of replicating this velocity laterally, creating a 16×8 *mesh of two-dimensional tensor product B-spline basis functions. This two-dimensional model was then inverted. We performed two full sweeps. In Figure 12.6.1 each symbol represents the residual RMS associated with one shot,*

$$\text{RMS} = \left(\frac{1}{m} \sum_{i=1}^{m} \left((t - t_i^c(\boldsymbol{\alpha}) \right)^2 \right)^{1/2},$$

measured in seconds. In the upper left panel of Figure 12.6.1 we see that the initial guess (*) is so poor that a number of shots do not produce enough arrivals and therefore they are absent in the inversion. The one-dimensional

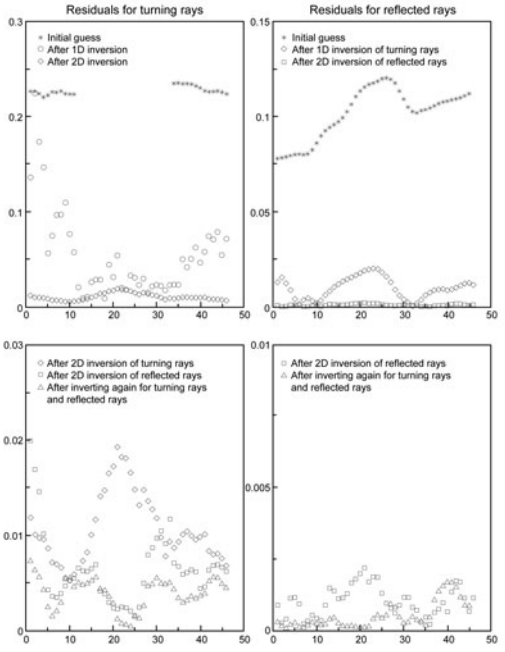
Residuals for turning rays
Initial guess
After 1D inversion
After 2D inversion
Residuals for reflected rays
Initial guess
After 1D inversion of turning rays
After 2D inversion of reflected rays
After 2D inversion of turning rays
After 2D inversion of reflected rays
After inverting again for turning rays and reflected rays
After 2D inversion of reflected rays
After inverting again for turning rays and reflected rays

Figure 12.6.1: RMS at different stages of inversion of the Silangkitang synthetic data set. We show RMS for the observations corresponding to each of 50 shots.

correction improves matters, especially in the middle of the model, but it is the two-dimensional lateral correction that brings the maximum RMS to about 17msec. For real, good quality data we would consider a good fit to have a maximum RMS under 10msec. In Figure 12.6.1 we cross-plot the RMS per shot for the two data sets at different stages of this process. Using the computed two-dimensional velocity as an initial guess, we invert the synthetic reflection data: that completes a full sweep through the whole data set. Now the maximum RMS has been reduced to below 3msec (lower-right figure), an excellent result that validates the approach, but, of course, remember that this is synthetic, noiseless data.

This page intentionally left blank

Appendix A

Sensitivity Analysis

A.1 Floating-point arithmetic

The classic reference for finite precision arithmetic is Wilkinson's monograph "Rounding Errors in Algebraic Processes" [255], while a more recent treatment is Higham's "Accuracy and Stability of Numerical Algorithms" [128]. Almost any numerical analysis book has an introductory chapter about this topic. Here we list some of the basic ideas used in our text.

Digital computers use floating-point representation for real and complex numbers based on the binary system, i.e., the basis is 2. Real numbers are rewritten in a special normalized form, where the mantissa is less than 1. Usually there is the option to use single (t-digit) or double ($2t$-digit) length mantissa representation and arithmetic. If we denote by $\mathbf{fl}(x)$ the floating-point computer representation of a real number x, and by $\oplus$ the floating-point addition, then the *unit round-off* μ (for a given computer) is defined as the smallest ε such that in floating-point arithmetic: $\mathbf{fl}(1) \oplus \varepsilon > \mathbf{fl}(1)$.

For a binary t-digit floating-point system $\mu = 2^{-t}$. The machine epsilon $\varepsilon_M = 2\mu$ is the gap between 1 and the next larger floating-point number (and, thus, in a relative sense, gives an indication of the gap between the floating-point numbers). Several of the bounds in this book contain the unit round-off or the machine precision; it is therefore advisable to check the size of ε_M for a particular machine and word length. A small Fortran program is available from Netlib to compute the machine precision for double precision and can be adapted easily for single.

Representation error. The relative error in the computer representation $\mathbf{fl}(x)$ of a real number $x \neq 0$ satisfies

$$\frac{|\mathbf{fl}(x) - x|}{|x|} \leq \mu,$$

implying that $\mathbf{fl}(x) \in [x(1-\varepsilon_M),\, x(1+\varepsilon_M)]$.

Rounding error. The error in a given floating-point operation $\circledast$, corresponding to the real operation $*$, satisfies

$$\mathbf{fl}(x) \circledast \mathbf{fl}(y) = (x * y)(1+\varepsilon), \qquad \text{with} \quad |\varepsilon| \leq \mu.$$

To measure the cost of the different algorithms described in the book we use as the unit the *flop*. A word of warning though: its definition differs from one author to the other; here we follow the one used in [105, 128], which is also common in many articles in the literature.

Definition 1. A flop is roughly the work associated with a floating-point operation (addition, subtraction, multiplication, or division).

In March 2011 the cheapest cost per Gigaflop (10^9 flops) was \$1.80, achieved on the computer cluster HPU4Science, made of six dual Core 2 Quad off-the-shelf machines at a cost of \$30,000, with performance enhanced by combining the CPUs with the Graphical PUs. In comparison, the cost in 1984 was \$15 million on a Cray X-MP.

A.2 Stability, conditioning and accuracy

A clear and concise review of these topics can be found in [57, 128, 237]. One general comment first: given a t-digit arithmetic, there is a limit to the attainable accuracy of any computation, because even the data themselves may not be representable by a t-digit number. Additionally, in practical applications, one should not lose sight of the fact that usually the data, derived from observations, already have a physical error much larger than the one produced by the floating-point representation.

Let us formally define a *mathematical problem* by a function that relates a data space $\mathcal{X}$ with a solutions space $\mathcal{Y}$, i.e., $P : \mathcal{X}(\text{data}) \to \mathcal{Y}(\text{solutions})$. Let us also define a specific *algorithm* for this problem as a function $\widetilde{P} : \mathcal{X} \to \mathcal{Y}$. One is interested in evaluating how close the solution computed by the algorithm is to the exact solution of the mathematical problem.

The accuracy will depend on the sensitivity of the mathematical problem P to perturbations of its data, the *condition of the problem*, and on the sensitivity of the algorithm $\tilde{P}$ to perturbations of the input data, the *stability* of the algorith.

The condition of the mathematical problem is commonly measured by the *condition number* $\kappa(\boldsymbol{x})$. We emphasize the problem dependency, so, for example, the same matrix A may give rise to an ill-conditioned least squares problem and a well-conditioned eigenvector problem.

A formal definition of the condition number follows.

Definition 2. The condition number is defined by

$$\kappa(\boldsymbol{x}) = \sup_{\delta \boldsymbol{x}} \frac{\|P(\boldsymbol{x}+\delta\boldsymbol{x}) - P(\boldsymbol{x})\|_2}{\|P(\boldsymbol{x})\|_2} \Big/ \frac{\|\delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}.$$

If the mapping P is differentiable with Jacobian $[J(\boldsymbol{x})]_{ij} = \frac{\partial P_i}{\partial x_j}$, the above formula can be replaced by

$$\kappa\left(\boldsymbol{x}\right) = \frac{\|J(\boldsymbol{x})\|_2 \, \|\boldsymbol{x}\|_2}{\|P\left(\boldsymbol{x}\right)\|_2}.$$

The condition number $\kappa\left(\boldsymbol{x}\right)$ is the leading coefficient of the data perturbation in the error bound for the solution.

Among the possible formalizations of the stability concept, *backward stability* is a convenient one; an algorithm is backward stable if it computes the exact solution for slightly perturbed data, or in other words, quoting [237], an algorithm is backward stable if "[it] gives the right answer to nearly the right question": $\widetilde{P}(\boldsymbol{x}) = P(\tilde{\boldsymbol{x}}) = P(\boldsymbol{x} + \Delta\boldsymbol{x})$. The perturbation of the input data $\|\Delta\mathbf{x}\|_2$ is the *backward error*. Formally,

Definition 3. The algorithm $\widetilde{P}$ is backward stable if the computed solution $\tilde{\boldsymbol{x}}$ is the exact solution of a slightly perturbed problem; i.e., if

$$\widetilde{P}(\boldsymbol{x}) = P(\tilde{\boldsymbol{x}})$$

for some $\widetilde{\mathbf{x}}$ with

$$\frac{\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = O(\varepsilon_M).$$

One can bound the actual error in the solution $P(\boldsymbol{x})$ of a problem with condition number $\kappa(\boldsymbol{x})$ if it is computed using a backward stable algorithm $\widetilde{P}\left(\boldsymbol{x}\right)$:

$$\frac{\left\|\widetilde{P}\left(\boldsymbol{x}\right) - P(\boldsymbol{x})\right\|_2}{\|P(\boldsymbol{x})\|_2} = O(\kappa(\boldsymbol{x})\varepsilon_M).$$

In words, a backward stable algorithm, when applied to a well-conditioned problem, yields an accurate solution, and if the backward error is smaller than the data errors, the problem is solved to the same extent that it is actually known. On the other hand, the computed solution to an ill-conditioned problem can have a very large error, even if the algorithm used was backward stable, the condition number acting as an amplification factor of the data errors.

This page intentionally left blank

Appendix B

Linear Algebra Background

B.1 Norms

Vector norms

The most commonly used norms for vectors are the l_1-, l_2- and l_∞-norms, denoted by $||\cdot||_1$, $||\cdot||_2$, and $||\cdot||_\infty$, respectively. These norms are defined by:

- $\|\boldsymbol{v}\|_1 = \sum_{i=1}^{n} |v_i|$.
- $\|\boldsymbol{v}\|_2 = \left(\sum_{i=1}^{n} |v_i|^2\right)^{\frac{1}{2}} = \left(\boldsymbol{v}^T\boldsymbol{v}\right)^{\frac{1}{2}}$ (Euclidean norm).
- $\|\boldsymbol{v}\|_\infty = \max_{1\le i\le n} |v_i|$ (Chebyshev norm).

A useful relationship between an inner product and the l_2-norms of its factors is the Cauchy-Schwartz inequality:

$$\left|\boldsymbol{x}^T\boldsymbol{y}\right| \le \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2 .$$

Norms are continuous functions of the entries of their arguments. It follows that a sequence of vectors $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$ converges to a vector $\boldsymbol{x}$ if and only if $\lim_{k\to\infty} \|\boldsymbol{x}_k - \boldsymbol{x}\| = 0$ for any norm.

Matrix norms

A natural definition of a matrix norm is the so-called induced or operator norm that, starting from a vector norm $\|\cdot\|$, defines the matrix norm as the maximum amount that the matrix A can stretch a unit vector, or more formally: $\|A\| = \max_{\|\boldsymbol{v}\|=1} \|A\boldsymbol{v}\|$. Thus, the induced norms associated with the usual vector norms are

- $\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$.
- $\|A\|_2 = [\max(\text{eigenvalue of } (A^T A))]^{\frac{1}{2}}$.
- $\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$.

In addition the so-called Frobenius norm (Euclidean length of A considered as an nm-vector) is

- $\|A\|_{\mathrm{F}} = \left(\sum_{j=1}^{n} \sum_{i=1}^{m} |a_{ij}|^2\right)^{\frac{1}{2}} = \mathrm{trace}(A^T A)^{\frac{1}{2}}$.

For square, orthogonal matrices $Q \in \mathbb{R}^{n \times n}$ we have $\|Q\|_2 = 1$ and $\|Q\|_{\mathrm{F}} = \sqrt{n}$. Both the Frobenius and the matrix l_2-norms are compatible with the Euclidean vector norm. This means that $\|A\boldsymbol{x}\| \leq \|A\| \, \|\boldsymbol{x}\|$ is true when using the l_2-norm for the vector and either the l_2 or the Frobenius norm for the matrix. Also, both are invariant with respect to orthogonal transformations Q:

$$\|QA\|_2 = \|A\|_2 , \qquad \|QA\|_{\mathrm{F}} = \|A\|_{\mathrm{F}} .$$

In terms of the singular values of A, the l_2 norm can be expressed as $\|A\|_2 = \max_i \sigma_i = \sigma_1$, where σ_i, $i = 1, \ldots, \min(m, n)$ are the singular values of A in descending order of size. In the special case of symmetric matrices the l_2-norm reduces to $\|A\|_2 = \max_i |\lambda_i|$, with λ_i an eigenvalue of A. This is also called the *spectral radius* of the matrix A.

B.2 Condition number

The condition number of a general matrix A in norm $\|\cdot\|_p$ is

$$\kappa_p(A) = \|A\|_p \left\|A^\dagger\right\|_p .$$

For a vector-induced norm, the condition number of A is the ratio of the maximum to the minimum stretch produced by the linear transformation represented by this matrix, and therefore it is greater than or equal to 1. In the l_2-norm, $\kappa_2(A) = \sigma_1/\sigma_r$, where σ_r is the smallest nonzero singular value of A and r is the rank of A.

In finite precision arithmetic, a large condition number can be an indication that the "exact" matrix is close to singular, as some of the zero singular values may be represented by very small numbers.

B.3 Orthogonality

The notation used for the inner product of vectors is $\boldsymbol{v}^T\boldsymbol{w} = \sum_i v_i w_i$. Note that $\|\boldsymbol{v}\|_2^2 = \boldsymbol{v}^T\boldsymbol{v}$. If $\boldsymbol{v}, \boldsymbol{w} \neq \mathbf{0}$, and $\boldsymbol{v}^T\boldsymbol{w} = 0$, then these vectors are orthogonal, and they are orthonormal if in addition they have unit length.

Orthogonal matrices. A square matrix Q is orthogonal if $Q^TQ = I$ or $QQ^T = I$, i.e., the columns or rows are orthonormal vectors and thus $\|Q\|_2 = 1$. It follows that orthogonal matrices represent isometric transformations that can only change the direction of a vector (rotation, reflection), but not its Euclidean norm, a reason for their practical importance: $\|Q\boldsymbol{v}\|_2 = \|\boldsymbol{v}\|_2$, $\|QA\|_2 = \|AQ\|_2 = \|A\|_2$.

Permutation matrix. A permutation matrix is an identity matrix with permuted rows or columns. It is orthogonal, and products of permutation matrices are again permutations.

Orthogonal projection onto a subspace of an inner product space. Given an orthonormal basis, $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n\}$ of a subspace $\mathcal{S} \subseteq \mathcal{X}$, where $\mathcal{X}$ is an inner product space, the orthogonal projection $\mathbf{P} : \mathcal{X} \to \mathcal{S}$ satisfies

$$\mathbf{P}\boldsymbol{x} = \sum_{i=1}^{n} (\boldsymbol{x}^T\boldsymbol{u}_i)\, \boldsymbol{u}_i.$$

The operator $\mathbf{P}$ is linear and satisfies $\mathbf{P}\boldsymbol{x} = \boldsymbol{x}$ if $\boldsymbol{x} \in \mathcal{S}$ (idempotent) and $\|\mathbf{P}\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_2\ \forall \boldsymbol{x} \in \mathcal{X}$. Therefore, the associated square matrix P is an orthogonal projection matrix if it is Hermitian and idempotent, i.e., if

$$P^T = P \text{ and } P^2 = P.$$

Note that an orthogonal projection divides the whole space into two orthogonal subspaces. If $\mathbf{P}$ projects a vector onto the subspace $\mathcal{S}$, then $\mathbf{I} - \mathbf{P}$ projects it onto $S^\perp$, the orthogonal complement of $\mathcal{S}$ with $S + S^\perp = \mathcal{X}$ and $S \cap S^\perp = 0$. An orthogonal projection matrix P is not necessarily an orthogonal matrix, but $I - 2P$ is orthogonal (see Householder transformations in Section 4.3).

An important projector is defined by a matrix of the form $P_U = UU^T$, where U has p orthonormal columns $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_p$. This is a projection onto the subspace spanned by the columns of U. In particular, the projection onto the subspace spanned by a single (not necessarily of norm 1) vector $\boldsymbol{u}$ is defined by the rank-one matrix $P_{\boldsymbol{u}} = \frac{\boldsymbol{u}\boldsymbol{u}^T}{\boldsymbol{u}^T\boldsymbol{u}}$.

Gram matrix. The Gram matrix or Grammian Gram(A) of an $m \times n$ matrix A is A^TA. Its elements are thus the n^2 possible inner products between pairs of columns of A.

B.4 Some additional matrix properties

The **Sherman-Morrison-Woodbury** formula gives a representation of the inverse of a rank-one perturbation of a matrix in terms of its inverse:

$$(A + \boldsymbol{u}\boldsymbol{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\boldsymbol{u}\boldsymbol{v}^T A^{-1}}{1 + \boldsymbol{v}^T A^{-1}\boldsymbol{u}},$$

provided that $1 + \boldsymbol{v}^T A^{-1}\boldsymbol{u} \neq 0$. As usual, for calculations, $A^{-1}\boldsymbol{u}$ is short hand for "solve the system" $A\boldsymbol{x} = \boldsymbol{u}$.

The next theorem presents the **interlacing property of the singular values** of A with those of matrices obtained by removing or adding a column or a row (see [20]).

Theorem 4. *Let A be bordered by a column $\boldsymbol{u} \in \mathbb{R}^m$, $\hat{A} = (A, \boldsymbol{u}) \in \mathbb{R}^{m\times n}$, $m \geq n$. Then, the ordered singular values σ_i of A separate the singular values of $\widehat{A}$ as follows:*

$$\widehat{\sigma}_1 \geq \sigma_1 \geq \widehat{\sigma}_2 \geq \sigma_2 \geq ... \geq \widehat{\sigma}_{n-1} \geq \sigma_{n-1} \geq \widehat{\sigma}_n.$$

Similarly, if A is bordered by a row $\boldsymbol{v} \in \mathbb{R}^n$,

$$\hat{A} = \begin{pmatrix} A \\ \boldsymbol{v}^T \end{pmatrix} \in \mathbb{R}^{m\times n}, \qquad m \geq n,$$

then

$$\widehat{\sigma}_1 \geq \sigma_1 \geq \widehat{\sigma}_2 \geq \sigma_2 \geq ... \geq \widehat{\sigma}_{n-1} \geq \sigma_{n-1} \geq \widehat{\sigma}_n \geq \sigma_n.$$

Appendix C

Advanced Calculus Background

C.1 Convergence rates

Definition 5. Let x^*, $x_k \in \mathbb{R}$ for $k = 0, 1, \ldots$ The sequence $\{x_k\}$ is said to converge to x^* if

$$\lim_{k \to \infty} |x_k - x^*| = 0.$$

The convergence is
linear if $\exists\ c \in [0, 1)$ and an integer $K > 0$ such that for $k \geq K$,

$$|x_{k+1} - x^*| \leq c\,|x_k - x^*|\,;$$

superlinear if $\exists\ c_k \longrightarrow 0$ and an integer $K > 0$ such that for $k \geq K$,

$$|x_{k+1} - x^*| \leq c_k\,|x_k - x^*|\,;$$

quadratic if $\exists\ c \in [0, 1)$ and an integer $K > 0$ such that for $k \geq K$,

$$|x_{k+1} - x^*| \leq c\,|x_k - x^*|^2\,.$$

Definition 6. A locally convergent iterative algorithm converges to the correct answer if the iteration starts close enough. **A globally convergent iterative algorithm** converges when starting from almost any point. For minimization, this is not to be confused with finding the global minimum of a functional on a compact domain (see below).

Global and local minimum: $\boldsymbol{x}^*$ is a global minimizer of a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ on a compact domain D if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathbb{R}^n$. $\boldsymbol{x}^*$ is a local minimizer inside a certain region, usually defined as an open "ball" of size δ around $\boldsymbol{x}^*$, if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for $\|\boldsymbol{x} - \boldsymbol{x}^*\|_2 < \delta$.

C.2 Multivariable calculus

The gradient and Hessian of a scalar function of several variables $f(\boldsymbol{x})$ are a vector and a matrix, respectively, defined by

$$\nabla f(\boldsymbol{x}) \equiv (\partial f/\partial x_1, ..., \partial f/\partial x_n)^T, \qquad \nabla^2 f(\boldsymbol{x}) \equiv \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right].$$

For a vector function of several variables

$$\boldsymbol{r}(\boldsymbol{x}) = (r_1(\boldsymbol{x}), r_2(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x}))^T,$$

with each $r_k(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$, we denote by $J(\boldsymbol{x})$ the Jacobian of $\boldsymbol{r}(\boldsymbol{x})$ and by G_k the Hessian of a component function r_k:

$$J(\boldsymbol{x}) = \left[\frac{\partial r_i}{\partial x_j}\right], \qquad G_k(\boldsymbol{x}) = \left[\frac{\partial^2 r_k}{\partial x_i \partial x_j}\right].$$

Definition 7. Descent direction $\boldsymbol{p}$ of a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$.

$\boldsymbol{p}$ is a descent direction at $\boldsymbol{x}_c$ for a function $f(\boldsymbol{x})$ if for sufficiently small and positive α $\|f(\boldsymbol{x}_c+\alpha\boldsymbol{p}\| < \|f(\boldsymbol{x}_c)\|$. Alternatively, $\boldsymbol{p}$ is a descent direction at $\boldsymbol{x}_c$ if the directional derivative (projection of the gradient on a given direction) of the function $f(\boldsymbol{x})$ at $\boldsymbol{x}_c$ in the direction $\boldsymbol{p}$ is negative: $\triangledown f(\boldsymbol{x}_c)^T\boldsymbol{p} < 0$.

Theorem 8. ***Taylor's theorem for a scalar function.***

If $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is continuously differentiable, then, for some $t \in [0, 1]$,

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x} + t\boldsymbol{p})^T\boldsymbol{p}.$$

If $f(\boldsymbol{x})$ is twice continuously differentiable then, for some $t \in [0, 1]$,

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T\boldsymbol{p} + \boldsymbol{p}^T\nabla^2 f(\boldsymbol{x} + t\boldsymbol{p})\boldsymbol{p}.$$

A necessary condition for $\boldsymbol{x}^*$ to be a stationary point of $f(\boldsymbol{x})$ is that $\nabla f(\boldsymbol{x}^*) = 0$. The sufficient conditions for $\boldsymbol{x}^*$ to be a local minimizer are $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ and $\nabla^2(\boldsymbol{x}^*)$ is positive definite.

The derivative $\mathbf{D}A(\boldsymbol{x})$ of an $m \times n$ nonlinear matrix function $A(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^k$ is a tri-dimensional tensor formed with k matrices (slabs) of dimension $m \times n$, each one containing the partial derivatives of the elements of A with respect to one of the variables of the vector $\boldsymbol{x}$. Thus, the second derivative of the vector function $\boldsymbol{r}(\boldsymbol{x})$ is the tri-dimensional tensor $\mathbf{G} = [\mathbf{G}_1, ..., \mathbf{G}_m]$.

The derivative of the orthogonal projector $P_{A(\boldsymbol{x})}$ onto the column space of a differentiable $m \times n$ matrix function $A(\boldsymbol{x})$ of local constant rank can be obtained as follows.

Lemma 9. (Lemma 4.1 in [101]) Let $A^\dagger(x)$ be the pseudoinverse of $A(\boldsymbol{x})$. Then $P_{A(\boldsymbol{x})} = AA^\dagger$ and

$$\mathbf{D}P_{A(\boldsymbol{x})} = P^\perp_{A(\boldsymbol{x})}\mathbf{D}AA^\dagger + (P^\perp_{A(\boldsymbol{x})}\mathbf{D}AA^\dagger)^T,$$

where $P^\perp_A = I - P_{A(\boldsymbol{x})}$.

Definition 10. A function f is Lipschitz continuous with constant γ in a set $X \subseteq \mathbb{R}$ if

$$\forall x,\, y \in X, \quad |f(x) - f(y)| \leq \gamma\,|x - y|\,. \tag{C.2.1}$$

Lipschitz continuity is an intermediary concept between continuity and differentiability. The operator $V : \mathbb{R}^n \to \mathbb{R}^n$, is Lipschitz continuous with constant γ in a set $X \subseteq \mathbb{R}^n$ if

$$\forall \boldsymbol{x},\, \boldsymbol{y} \in X, \quad \|V(\boldsymbol{x}) - V(\boldsymbol{y})\|_2 \leq \gamma\,\|\boldsymbol{x} - \boldsymbol{y}\|_2\,.$$

V is contracting if it is Lipschitz continuous with constant less than unity: $\gamma < 1$. V is uniformly monotone if there exists a positive m such that

$$m\,\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \leq (V(\boldsymbol{x}) - V(\boldsymbol{y}))^T(\boldsymbol{x} - \boldsymbol{y}).$$

V is vector Lipschitz if

$$|V(\boldsymbol{x}) - V(\boldsymbol{y})| \leq A\,|\boldsymbol{x} - \boldsymbol{y}|\,,$$

where A is a non-negative $n \times n$ matrix and the inequality is meant elementwise. V is vector contracting if it is vector Lipschitz continuous with $\|A\| < 1$.

Lagrange multipliers

Lagrange multipliers are used to find the local extrema of a function $f(\boldsymbol{x})$ of n variables subject to k equality constraints $g_i(\boldsymbol{x}) = c_i$, by reducing the problem to an $(n-k)$-variable problem without constraints. Formally, new scalar variables $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_k)^T$, one for each contraint, are introduced and a new function is defined as

$$F(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{k} \lambda_i(g_i(\boldsymbol{x}) - c_i).$$

The local extrema of this extended function F (the Lagrangian), occur at the points where its gradient is zero: $\nabla F(\boldsymbol{x}, \boldsymbol{\lambda}) = \mathbf{0}$, $\boldsymbol{\lambda} \neq \mathbf{0}$, or equivalently,

$$\nabla_x F(\boldsymbol{x}, \boldsymbol{\lambda}) = \mathbf{0}, \qquad \nabla_{\boldsymbol{\lambda}} F(\boldsymbol{x}, \boldsymbol{\lambda}) = \mathbf{0}.$$

This form encodes compactly the constraints, because

$$\nabla_{\lambda_i} F(\boldsymbol{x}, \boldsymbol{\lambda}) = 0 \Leftrightarrow g_i(\boldsymbol{x}) = c_i.$$

An alternative to this method is to use the equality constraints to eliminate k of the original variables. If the problem is large and sparse, this elimination may destroy the sparsity and thus the Lagrange multipliers approach is preferable.

Appendix D

Statistics

D.1 Definitions

We list some basic definitions and techniques taken from statistics that are needed to understand some sections of this book.

- A ***sample space*** is the set of possible outcomes of an experiment or a random trial. For some kind of experiments (trials), there may be several plausible sample spaces available. The complete set of outcomes can be constructed as a Cartesian product of the individual sample spaces. An ***event*** is a subset of a sample space.

- We denote by Pr{event} the ***probability of an event***. We often consider finite sample spaces, such that each outcome has the same probability (equiprobable).

- X is a ***random variable*** defined on a sample space if it assigns a unique numerical value to every outcome, i.e., it is a real-valued function defined on a sample space.

- X is a ***continuous random variable*** if it can assume every value in an interval, bounded or unbounded (continuous distribution). It can be characterized by a ***probability density function (pdf)*** $p(x)$ defined by
$$\Pr\{a \leq X \leq b\} = \int_a^b p(x)\,dx.$$

- X is a ***discrete random variable*** if it assumes at most a countable set of different values (discrete distribution). It can be characterized by its ***probability function (pf),*** which specifies the probability

that the random variable takes each of the possible values from say $\{x_1, x_2, \ldots, x_i, \ldots\}$:

$$p(x_i) = \Pr\{X = x_i\}.$$

- All distributions, discrete or continuous, can be characterized through their (cumulative) ***distribution function***, the total probability up to, and including, a point x. For a discrete random variable,

$$P(x_i) = \Pr\{X \leq x_i\}.$$

- The ***expectation*** or ***expected value*** $\mathcal{E}[X]$ for a random variable X or equivalently, the mean μ of its distribution, contains a summary of the probabilistic information about X. For a continuous variable X, the expectation is defined by

$$\mathcal{E}[X] = \int_{-\infty}^{\infty} x\, p(x)\, dx.$$

For a discrete random variable with probability function $p(x)$ the expectation is defined as

$$\mathcal{E}[X] = \sum_{\forall i} x_i\, p(x_i).$$

For a discrete random variable X with values $x_1, x_2, \ldots, x_N$ and with all $p(x_i)$ equal (all x_i equiprobable, $p(x_i) = \frac{1}{N}$), the expected value coincides with the ***arithmetic mean***

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i.$$

- The expectation is linear, i.e., for a, b, c constants and X, Y two random variables,

$$\begin{aligned} \mathcal{E}[X + c] &= \mathcal{E}[X] + c, \\ \mathcal{E}[aX + bY] &= \mathcal{E}[aX] + \mathcal{E}[bY]. \end{aligned}$$

- A convenient measure of the dispersion (or spread about the average) is the ***variance*** of the random variable, $\mathrm{var}[X]$ or ς^2:

$$\mathrm{var}[X] = \varsigma^2 = \mathcal{E}[(X - \mathcal{E}(X))^2].$$

In the equiprobable case this reduces to

$$\mathrm{var}[X] = \varsigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2.$$

- For a random sample of observations $x_1, x_2, \ldots, x_N$, the following formula is an estimate of the variance ς^2:

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2.$$

  In the case of a sample from a normal distribution this is a particularly good estimate.

- The square root of the variance is the ***standard deviation,*** ς. It is known by physicists as RMS or ***root mean square*** in the equiprobable case:

$$\varsigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}.$$

- Given two random variables X and Y with expected values $\mathcal{E}[X]$ and $\mathcal{E}[Y]$, respectively, the ***covariance*** is a measure of how the values of X and Y are related:

$$\mathrm{Cov}\{X, Y\} = \mathcal{E}(XY) - \mathcal{E}(X)\mathcal{E}(Y).$$

  For random vectors $X \in \mathbb{R}^m$ and $Y \in \mathbb{R}^n$ the covariance is the $m \times n$ matrix

$$\mathrm{Cov}\{X, Y\} = \mathcal{E}(XY^T) - \mathcal{E}(X)E(X)^T.$$

  The (i, j)th element of this matrix is the covariance between the ith component of X and the jth component of Y.

- In order to estimate the degree of interrelation between variables in a manner not influenced by measurement units, the (Pearson) ***correlation coefficient*** is used:

$$c_{XY} = \frac{\mathrm{Cov}\{X, Y\}}{(\mathrm{var}[X]\,\mathrm{var}[Y])^{\frac{1}{2}}}.$$

  Correlation is a measure of the strength of the linear relationship between the random variables; nonlinear ones are not measured satisfactorily.

- If $\mathrm{Cov}\{X, Y\} = 0$, the correlation coefficient is zero and the variables X, Y are ***uncorrelated***.

- The random variables X and Y, with distribution functions $P_X(x)$, $P_Y(y)$ and densities $p_X(x)$, $p_Y(y)$, respectively, are statistically ***independent*** if and only if the combined random variable (X, Y) has a joint cumulative distribution function $P_{X,Y}(x, y) = P_X(x)P_Y(y)$ or

equivalently a joint density $p_{X,Y}(x,y) = p_X(x)p_Y(y)$. The expectation and variance operators have the properties $\mathcal{E}[XY] = \mathcal{E}[X]\,\mathcal{E}[Y]$, $\text{var}[X+Y] = \text{var}[X] + \text{var}[Y]$. It follows that independent random variables have zero covariance.

- Independent variables are uncorrelated, but the opposite is not true unless the variables belong to a normal distribution.

- A random vector $\boldsymbol{w}$ is a ***white noise*** vector if $\mathcal{E}(\boldsymbol{w}) = 0$ and $\mathcal{E}(\boldsymbol{w}\boldsymbol{w}^T) = \varsigma^2 I$. That is, it is a zero mean random vector where all the elements have identical variance. Its autocorrelation matrix is a multiple of the identity matrix; therefore the vector elements are uncorrelated. Note that Gaussian noise $\nLeftrightarrow$ white noise.

- The ***coefficient of variation*** is a normalized measure of the dispersion:
$$c_v = \frac{\varsigma}{\mu}.$$
In signal processing the reciprocal ratio $\frac{\mu}{\varsigma}$ is referred to as the signal to noise ratio.

- The ***coefficient of determination*** R^2 is used in the context of linear regression analysis (statistical modeling), as a measure of how well a linear model fits the data. Given a model M that predicts values M_i, $i = 1, \ldots, N$ for the observations $x_1, x_2, \ldots, x_N$ and the residual vector $\boldsymbol{r} = (M_1 - x_1, \ldots, M_N - x_N)^T$, the most general definition for the coefficient of determination is
$$R^2 = 1 - \frac{\|\boldsymbol{r}\|_2^2}{\sum_{i=1}^N (x_i - \overline{x})^2}.$$
In general, it is an approximation of the unexplained variance, since the second term compares the variance in the model's errors with the total variance of the data $x_1, \ldots, x_N$.

- The ***normal (or Gaussian)*** probability function $N(\mu, \varsigma^2)$ has mean μ and variance ς^2. Its probability density function takes the form
$$p(x) = \frac{1}{\sqrt{2\pi}\varsigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\varsigma}\right)^2\right].$$

- If X_i, $i = 1, \ldots, n$ are random variables with normal distributions $N(0,1)$, then $\sum_{i=1}^n X_i^2$ has a chi-square distribution with n degrees of freedom. Given two independent random variables Y, W with chi-square distributions with m and n degrees of freedom, respectively,

the random variable X,

$$X = \frac{Y/m}{W/n},$$

has an ***F-distribution*** with m and n degrees of freedom, $F(m, n)$.

- The ***Student's*** t_n random variable with n degrees of freedom is defined as

$$t_n = \frac{Z}{\sqrt{\chi_n^2/n}},$$

where Z is a standard normal random variable, χ_n^2 is a chi-square random variable with n degrees of freedom and Z and χ_n^2 are independent.

- The F-distribution becomes relevant when testing hypotheses about the variances of normal distributions. For example, assume that we have two independent random samples (of sizes n_1 and n_2 respectively) from two normal distributions; the ratio of the unbiased estimators s_1 and s_2 of the two variances,

$$\frac{s_1/n_1 - 1}{s_2/n_2 - 1},$$

is distributed according to an F-distribution $F(n_1, n_2)$ if the variances are equal: $\varsigma_1^2 = \varsigma_2^2$.

- A ***time series*** is an ordered sequence of observations. The ordering is usually in time, often in terms of equally spaced time intervals, but it can also be ordered in, for example, space. The time series elements are frequently considered random variables with the same mean and variance.

- A ***periodogram*** is a data analysis technique for examining the frequency domain of an equispaced time series and search for hidden periodicities. Given a time series vector of N observations $x(j)$, the discrete Fourier transform (DFT) is given by a complex vector of length N:

$$X(k) = \sum_{j=1}^{N} x(j)\varpi_N^{(j-1)(k-1)},\ 1 \leq k \leq N,$$

with $\varpi_N = \exp((-2\pi i)/N)$.

- The magnitude squared of the discrete Fourier transform components $|X(k)|^2$ is called the power. The periodogram is a plot of the power components versus the frequencies $\{\frac{1}{N}, \frac{2}{N}, ...\frac{k}{N}, ...\}$.

D.2 Hypothesis testing

In ***hypothesis testing*** one uses statistics to determine whether a given hypothesis is true. The process consists of four steps:

- Formulate the null hypothesis H_0 and the alternative hypothesis H_a, which are mutually exclusive.
- Identify a test statistics t that can be used to assess the truth of the null hypothesis from the sample data. It can involve means, proportions, standard deviations, etc.
- Determine the corresponding distribution function, assuming that the null hypothesis is true and, after choosing an acceptable ***level of significance*** α (common choices are 0.01, 0.05, 0.1), determine the critical region for t (see details below).
- Compute the t for the observations. If the computed value of the test statistics falls into the critical region, reject H_0.

In other words, the test of a hypothesis on the significance level α is performed by means of a statistic t and critical values $\underline{t}_\alpha$ and $\bar{t}_\alpha$ so that

$$1 - \alpha = \Pr\left\{\underline{t}_\alpha \leq t \leq \bar{t}_\alpha\right\}, \qquad 0 \leq \alpha \leq 1$$

if H_0 holds. The hypothesis H_0 is rejected if t falls outside the range $[\underline{t}_\alpha, \bar{t}_\alpha]$. This guarantees that H_0 will only be erroneously rejected in $100 \times \alpha\%$ of the cases.

For example, assume that we have approximated m observations $y_1, y_2, \ldots, y_m$ with a linear model M_{full} with n terms and we want to know if k of these terms are redundant, i.e., if one could get a good enough model when setting k coefficient of the original model to 0. We have the residual norms ρ_{full} and ρ_{red} for both possible models, with $\rho_{\text{full}} < \rho_{\text{red}}$.

We formulate the hypothesis H_0: the reduced model is good enough, i.e., the reduction in residual when using all n terms is negligible.

Under the assumption that the errors in the data $y_1, y_2, \ldots, y_m$ are normally distributed with constant variance and zero mean, we can choose a statistic based on a proportion of variance estimates:

$$f_{\text{obs}} = \frac{\rho_{\text{red}} - \rho_{\text{full}}}{\rho_{\text{full}}} \frac{m-n}{k}.$$

This statistic follows an F-distribution with k and $m-n$ degrees of freedom: $F(k, m-n)$. The common practice is to denote by f_α the value of the statistic with cumulative probability $F_\alpha(k, m-n) = 1 - \alpha$.

If $f_{\text{obs}} > f_\alpha$, the computed statistic falls into the critical region for the F-distribution and the H_0 hypothesis should be rejected, with a possible error of α, i.e., we do need all the terms of the full model.

References

[1] J. J. Alonso, I. M. Kroo and A. Jameson, "Advanced algorithms for design and optimization of quiet supersonic platforms." AIAA Paper 02-0114, 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, 2002.

[2] J. J. Alonso, P. Le Gresley, E. van der Weide and J. Martins, "pyMDO: a framework for high-fidelity multidisciplinary optimization." 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, 2004.

[3] A. Anda and H. Park, "Fast plane rotations with dynamic scaling." SIAM J. Matrix Anal. Appl. **15**:162–174, 1994.

[4] E. Angelosante, D. Giannakis and G. B. Grossi, "Compressed sensing of time-varying signals." Digital Signal Processing, 16th International Conference, Santorini, Greece, 2009.

[5] H. Ashley and M. Landahl, *Aerodynamics of Wings and Bodies*. Dover, New York, 1985.

[6] I. Barrodale and F. D. K. Roberts, "An improved algorithm for discrete ℓ_1 linear approximation." SIAM J. Numer. Anal. **10**:839–848, 1973.

[7] R. Bartels, J. Beaty and B. Barski, *An Introduction to Splines for Use in Computer Graphics and Parametric Modeling*. Morgan Kaufman Publishers, Los Altos, CA, 1987.

[8] A. E. Beaton and J. W. Tukey, "The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data." Technometrics **16**:147–185, 1974.

[9] D. M. Bates and D. G. Watts, *Nonlinear Regression Analysis and Its Applications*. J. Wiley, New York, 1988.

[10] G. M. Baudet, "Asynchronous iterative methods for multiprocessors." J. ACM **15**:226–244, 1978.

[11] A. Beck, A. Ben-Tal and M. Teboulle, "Finding a global optimal solution for a quadratically constrained fractional quadratic problem with applications to the regularized total least squares." SIAM J. Matrix Anal. Appl. **28**:425–445, 2006.

[12] M. Berry, "Large scale sparse singular value computations." International J. Supercomp. Appl. **6**:13–49, 1992.

[13] P. R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences*. McGraw-Hill, New York, 1969.

[14] C. H. Bischof and G. Qintana-Orti, "Computing rank-revealing QR factorization of dense matrices." ACM TOMS **24**:226–253, 1998.

[15] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[16] Å. Björck, "Solving linear least squares problems by Gram-Schmidt orthogonalization." BIT **7**:1–21, 1967.

[17] Å. Björck, "Iterative refinement of linear least squares solutions." BIT **7**:257–278, 1967.

[18] Å. Björck, "Iterative refinement of linear least squares solutions II." BIT **8**:8–30, 1968.

[19] Å. Björck, "Stability analysis of the method of seminormal equations for linear least squares problems." Lin. Alg. Appl. **88/89**:31–48, 1987.

[20] Å. Björck, *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

[21] Å. Björck, "The calculation of linear least squares problems." Acta Numerica **13**:1–53, 2004.

[22] Å. Björck and I. S. Duff, "A direct method for the solution of sparse linear least squares problems." Lin. Alg. Appl. **34**:43–67, 1980.

[23] Å. Björck and G. H. Golub, "Iterative refinement of linear least squares solution by Householder transformation." BIT **7**:322–337, 1967.

[24] Å. Björck, E. Grimme and P. Van Dooren, "An implicit shift bidiagonalization algorithm for ill-posed systems." BIT **34**:510–534, 1994.

[25] Å. Björck, P. Heggernes and P. Matstoms, "Methods for large scale total least squares problems." SIAM J. Matrix Anal. Appl. **22**:413–429, 2000.

[26] Å. Björck and C. C. Paige, "Loss and recapture of orthogonality in the modified Gram-Schmidt algorithm." SIAM J. Matrix Anal. **13**:176–190, 1992.

[27] Å. Björck and V. Pereyra, "Solution of Vandermonde systems of equations." Math. Comp. **24**:893–904, 1970.

[28] C. Böckman, "A modification of the trust-region Gauss-Newton method for separable nonlinear least squares problems." J. Math. Systems, Estimation and Control **5**:1–16, 1995.

[29] C. de Boor, *A Practical Guide to Splines*. Applied Mathematical Sciences **27**. Springer, New York, 1994.

[30] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2003.

[31] R. Brent, *Algorithms for Minimization Without Derivatives*. Prentice Hall, Englewood Cliffs, NJ, 1973. Reprinted by Dover, New York, 2002.

[32] D. Calvetti, P. C. Hansen and L. Reichel, "L-curve curvature bounds via Lanczos bidiagonalization." Electr. Transact. on Num. Anal. **14**:134–149, 2002.

[33] D. Calvetti, G. H. Golub and L. Reichel, "Estimation of the L-curve via Lanczos bidiagonalization algorithm for ill-posed systems." BIT **39**:603–619, 1999.

[34] E. J. Candes, "Compressive sampling." Proceedings of the International Congress of Mathematicians, Madrid, Spain, 2006.

[35] E. J. Candes and M. B. Wakin, "People hearing without listening: An introduction to compressive sampling." Signal Processing Magazine, IEEE **25**:21–30, 2008.

[36] L. Carcione, J. Mould, V. Pereyra, D. Powell and G. Wojcik, "Nonlinear inversion of piezoelectrical transducer impedance data." J. Comp. Accoustics **9**:899–910, 2001.

[37] L. Carcione, V. Pereyra and D. Woods, *GO: Global Optimization*. Weidlinger Associates Report, 2005.

[38] R. I. Carmichael and L. I. Erickson, "A higher order panel method for predicting subsonic or supersonic linear potential flow about arbitrary configurations." American Institute of Aeronautics and Astronautics Paper 81–1255, 1981.

[39] M. Chan, "Supersonic aircraft optimization for minimizing drag and sonic boom." Ph.D. Thesis, Stanford University, Stanford, CA, 2003.

[40] T. F. Chan, "Rank revealing QR-factorizations." Lin. Alg. Appl. **88/89**:67–82, 1987.

[41] T. F. Chan and D. E. Foulser, "Effectively well-conditioned linear systems." SIAM J. Sci. Stat. Comput. **9**:963–969, 1988.

[42] T. F. Chan and P. C. Hansen, "Computing truncated SVD least squares solutions by rank revealing QR-factorizations." SIAM J. Sci. Statist. Comput. **11**:519–530, 1991.

[43] D. Chazan and W. L. Miranker, "Chaotic relaxation." Lin. Alg. Appl. **2**:199–222, 1969.

[44] W. Cheney and D. Kincaid, *Numerical Mathematics and Computing.* Brooks/Cole, Belmont, CA, 2007.

[45] S. Choi, J. J. Alonso and H. S. Chung, "Design of low-boom supersonic business jet using evolutionary algorithms and an adaptive unstructured mesh method." 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Palm Springs, CA, 2004.

[46] H. S. Chung, S. Choi and J. J. Alonso, "Supersonic business jet design using knowledge-based genetic algorithms with adaptive, unstructured grid methodology." AIAA 2003-3791, 21st Applied Aerodynamic Conference, Orlando, Fl., June 2003.

[47] H. S. Chung, "Multidisciplinary design optimization of supersonic business jets using approximation model-based genetic algorithms." Ph.D. Thesis, Stanford University, Stanford, CA, 2004.

[48] J. F. Claerbout and F. Muir, "Robust modeling with erratic data." Geophysics **38**:826–844, 1973.

[49] A. K. Cline, "An elimination method for the solution of linear least squares problems." SIAM J. Numer. Anal. **10:**283–289, 1973.

[50] D. Coleman, P. Holland, N. Kaden, V. Klema and S. C. Peters, "A system of subroutines for iteratively reweighted least squares computations." ACM TOMS **6**:328–336, 1980.

[51] T. F. Coleman and Y. Li, "A globally and quadratically convergent affine scaling method for linear ℓ_1 problems." Mathematical Programming, Series A **56**:189–222, 1992.

[52] T. P. Collignon, "Efficient iterative solution of large linear systems on heterogeneous computing systems." Ph. D. Thesis, TU Delft, The Netherlands, 2011.

[53] T. P. Collignon and M. B. van Gijzen. "Parallel scientific computing on loosely coupled networks of computers." In B. Koren and C. Vuik, editors, *Advanced Computational Methods in Science and Engineering*. Springer Series Lecture Notes in Computational Science and Engineering, **71**:79–106. Springer Verlag, Berlin/Heidelberg, Germany, 2010.

[54] G. Cybenko, "Approximation by superpositions of a sigmoidal function." Math. Control Signals Systems **2**:303–314, 1989.

[55] J. Dahl, P. C. Hansen, S. H. Jensen and T. L. Jensen, "Algorithms and software for total variation image reconstruction via first-order methods." Numer. Algo. **53**:67–92, 2010.

[56] J. Dahl and L. Vanderberghe, CVXOPT: *A Python Package for Convex Optimization.* http://abel.ee.ucla.edu/cvxopt, 2012.

[57] G. Dahlquist and Å. Björck, *Numerical Methods in Scientific Computing.* SIAM, Philadelphia, 2008.

[58] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection." ACM TOMS **38**:1–25, 2011.

[59] K. Deb, A. Pratap, S. Agrawal and T. Meyarivan, *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II.* Technical Report No. 2000001. Indian Institute of Technology, Kanpur, India, 2000.

[60] P. Deift, J. Demmel, L.-C. Li and C. Tomei, "The bidiagonal singular value decomposition and Hamiltonian mechanics." SIAM J. Numer. Anal. **28**:1463–1516, 1991.

[61] R. S. Dembo, S. C. Eisenstat and T. Steihaug, "Inexact Newton methods." SIAM J. Numer. Anal. **19**:400–408, 1982.

[62] C. J. Demeure and L. L. Scharf, "Fast least squares solution of Vandermonde systems of equations." Acoustics, Speech and Signal Processing **4**:2198–2210, 1989.

[63] J. W. Demmel, *Applied Numerical Linear Algebra.* SIAM, Philadelphia, 1997.

[64] J. Demmel, Y. Hida, W. Riedy and X. S. Li, "Extra-precise iterative refinement for overdetermined least squares problems." ACM TOMS **35**:1–32, 2009.

[65] J. Dennis, D. M. Gay and R. Welsch, "Algorithm 573 NL2SOL – An adaptive nonlinear least-squares algorithm." ACM TOMS **7**:369–383, 1981.

[66] J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* SIAM, Philadelphia, 1996.

[67] J. E. Dennis and H. F. Walker, "Convergence theorems for least-change secant update methods." SIAM J. Num. Anal. **18**:949–987, 1981.

[68] A. P. Dempster, N. M. Laird and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm (with discussion)." Journal of the Royal Statistical Society B **39**:1–38, 1977.

[69] P. Dierckx, *Curve and Surface Fitting with Splines.* Clarendon Press, Oxford, 1993.

[70] J. Dongarra, J. R. Bunch, C. B. Moler and G. W. Stewart, *LINPACK User's Guide.* SIAM, Philadelphia, 1979.

[71] N. Draper and H. Smith, *Applied Regression Analysis.* J. Wiley, New York, 1981.

[72] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank." Psychometrika **1**:211–218, 1936.

[73] L. Eldén, "A note on the computation of the generalized cross-validation function for ill-conditioned least squares problems." BIT **24**:467–472, 1984.

[74] L. Eldén, "Perturbation theory for the linear least squares problem with linear equality constraints." BIT **17**:338–350, 1980.

[75] L. Eldén, *Matrix Methods in Data Mining and Pattern Recognition.* SIAM, Philadelphia, 2007.

[76] J. Eriksson, P.-Å. Wedin, M. E. Gulliksson and I. Söderkvist, "Regularization methods for uniformly rank-deficient nonlinear least-squares problems." J. Optimiz. Theory and Appl. **127**:1–26, 2005.

[77] J. Eriksson and P.-Å. Wedin, "Truncated Gauss-Newton algorithms for ill-conditioned nonlinear least squares problems." Optimiz. Meth. and Software **19**:721–737, 2004.

[78] R. D. Fierro, P. C. Hansen and P. S. K. Hansen, "UTV tools: MATLAB templates for rank-revealing UTV decompositions." Numer. Algo. **20**:165–194, 1999. The software is available from: htpp://www.netlib.org/numeralgo.

[79] P. M. Fitzpatrick, *Advanced Calculus.* Second edition Brooks/Cole, Belmont, CA, 2006.

[80] D. Fong and M. A. Saunders, "LSMR: An iterative algorithm for sparse least-squares problems." SIAM J. Sci. Comput. **33**:2950–2971, 2011.

[81] G. E. Forsythe, "Generation and use of orthogonal polynomials for data-fitting with a digital computer." J. SIAM **5**:74–88, 1957.

[82] J. Fox and S. Weisberg, *Robust Regression in R*, An Appendix to *An R Companion to Applied Regression*, Second edition. http://socserv.socsci.mcmaster.ca/jfox/Books /Companion/appendix.html.

[83] C. F. Gauss, *Theory of the Combination of Observations Least Subject to Errors. Parts 1 and 2, Supplement,* G. W. Stewart. SIAM, Philadelphia, 1995.

[84] D. M. Gay, *Usage Summary for Selected Optimization Routines.* AT&T Bell Labs. Comp. Sc. Tech. Report, 1990.

[85] D. M. Gay and L. Kaufman, *Tradeoffs in Algorithms for Separable Nonlinear Least Squares.* AT&T Bell Labs. Num. Anal. Manuscript, 90–11, 1990.

[86] D. M. Gay, *NSF and NSG; PORT Library.* AT&T Bell Labs. http://www.netlib.org/port, 1997.

[87] P. E. Gill, G. H. Golub, W. Murray and M. A. Saunders, "Methods for modifying matrix factorisations." Math. Comp. **28**:505–535, 1974.

[88] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders and M. H. Wright, *Users Guide for LSSOL (Version 1.0): A Fortran Package for Constrained Linear Least-Squares and Convex Quadratic Programming.* Report 86-1 Department of Operation Research, Stanford University, CA, 1986.

[89] P. E. Gill, W. Murray and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization." SIAM Rev. **47**:99–131, 2005.

[90] P. E. Gill, W. Murray, M. A. Saunders and M. H. Wright, "Maintaining LU factors of a general sparse matrix." Linear Algebra and its Applications **88–89**:239–270, 1987.

[91] P. E. Gill, W. Murray and M. H. Wright, *Practical Optimization.* Academic Press, 1981.

[92] G. H. Golub, "Numerical methods for solving linear least squares problems." Numer. Math. **7**:206–16, 1965.

[93] G. H. Golub, P. C. Hansen and D. P. O'Leary, "Tikhonov regularization and total least squares." SIAM J. Matrix Anal. Appl. **21**:185–194, 1999.

[94] G. H. Golub, M. Heath and G. Wahba, "Generalized cross-validation as a method for choosing a good ridge parameter." Technometrics **21**:215–223, 1979.

[95] G. H. Golub, A. Hoffman and G. W. Stewart, "A generalization of the Eckhard-Young-Mirsky matrix approximation theorem." Linear Algebra Appl. **88/89**:317–327, 1987.

[96] G. H. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix." SIAM J, Numer. Anal. Ser. B **2**:205–224, 1965.

[97] G. H. Golub, V. Klema and G. W. Stewart, *Rank Degeneracy and Least Squares.* Report TR-456, Computer Science Department, University of Maryland, College Park, 1977.

[98] G. H. Golub and R. Le Veque, "Extensions and uses of the variable projection algorithm for solving nonlinear least squares problems." Proceedings of the Army Numerical Analysis and Computers Conference, White Sands Missile Range, New Mexico, pp. 1–12, 1979.

[99] G. H. Golub and U. von Matt, "Tikhonov regularization for large problems." Workshop on Scientific Computing, Ed. G. H. Golub, S. H. Lui, F. Luk and R. Plemmons, Springer, New York, 1997.

[100] G. H. Golub and V. Pereyra, *The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate.* STAN-CS-72-261, Stanford University, Computer Sciences Department, 1972. (It contains the original VARPRO computer code.)

[101] G. H. Golub and V. Pereyra, "The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate." SIAM J. Numer. Anal. **10**:413–432, 1973.

[102] G. H. Golub and V. Pereyra, "Differentiation of pseudoinverses, separable nonlinear least squares problems, and other tales." Proceedings MRC Seminar on Generalized Inverses and Their Applications, Ed. Z. Nashed, pp. 302–324, Academic Press, NY, 1976.

[103] G. H. Golub and V. Pereyra, "Separable nonlinear least squares: The variable projection method and its applications." Inverse Problems **19**:R1–R26, 2003.

[104] G. H. Golub and J. M. Varah, "On a characterization of the best ℓ_2-scaling of a matrix." SIAM J. Numer. Anal. **11**:472–279, 1974.

[105] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Third edition, John Hopkins University Press, Baltimore, 1996.

[106] G. H. Golub and C. F. Van Loan, "An analysis of the total least squares problem." SIAM J. Numer. Anal. **17**:883–893, 1980.

[107] G. H. Golub and J. H. Wilkinson, "Note on iterative refinement of least squares solutions." Numer. Math. **9**:139–148, 1966.

[108] J. F. Grcar, *Optimal Sensitivity Analysis of Linear Least Squares.* Lawrence Berkeley National Laboratory, Report LBNL-52434, 2003.

[109] J. F. Grcar, "Mathematicians of Gaussian elimination." Notices of the AMS **58**:782–792, 2011.

[110] J. F. Grcar, "John von Neumann's analysis of Gaussian elimination and the origins of modern Numerical Analysis." SIAM Review **53**:607–682, 2011.

[111] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* Other Titles in Applied Mathematics **105**. Second edition, SIAM, Philadelphia, 2008.

[112] E. Grosse, "Tensor spline approximation." Linear Algebra Appl. **34**:29–41, 1980.

[113] I. Gutman, V. Pereyra and H. D. Scolnik, "Least squares estimation for a class of nonlinear models." Technometrics **15**:209–218, 1973.

[114] Y. Y. Haimes, L. S. Ladon and D. A. Wismer, "On a bicriterion formulation of the problem of integrated system identification and system optimization." IEEE Trans. on Systems, Man and Cybernetics **1**:296–297, 1971.

[115] S. Hammarling, "A note on modifications to the Givens plane rotations." J. Inst. Math. Applic. **13**:215–218, 1974.

[116] S. Hammarling, *A Survey of Numerical Aspects of Plane Rotations.* University of Manchester, MIMS Eprint 2008.69, 1977.

[117] M. Hanke and P. C. Hansen, "Regularization methods for large-scale problems." Surv. Math. Ind. **3**:253–315, 1993.

[118] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems.* SIAM, Philadelphia, 1998.

[119] P. C. Hansen, "The L-curve and its use in the numerical treatment of inverse problems." In *Comp. Inverse Problems in Electrocardiology.* Ed. P. Johnston, pp. 119–142, WIT Press, Southampton, UK, 2001.

[120] P. C. Hansen, "Regularization tools version 4.0 for MATLAB 7.3." Numer. Algorithms **46**:189–194, 2007.

[121] P. C. Hansen, *Discrete Inverse Problems – Insight and Algorithms.* SIAM, Philadelphia, 2010.

[122] P. C. Hansen, M. Kilmer and R. H. Kjeldsen, "Exploiting residual information in the parameter choice for discrete ill-posed problems." BIT **46**:41–59, 2006.

[123] P. C. Hansen and M. Saxild-Hansen, "AIR tools – A MATLAB package of algebraic iterative reconstruction methods." J. Comp. Appl. Math. **236**:2167–2178, 2011.

[124] P. C. Hansen and P. Yalamov, "Computing symmetric rank-revealing decompositions via triangular factorization." SIAM J. Matrix Anal. Appl. **23**:443–458, 2001.

[125] J. G. Hayes, ed., *Numerical Approximation to Functions and Data.* Athlone Press, London, 1970.

[126] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems." J. Res. Nat. Stan. B. **49**:409–432, 1952.

[127] N. Higham, "Analysis of the Cholesky decomposition of a semi-definite matrix." In *Reliable Numerical Computing*, ed. M. G. Cox and S. J. Hammarling, Oxford University Press, London, 1990.

[128] N. Higham, *Accuracy and Stability of Numerical Algorithms.* Second edition, SIAM, Philadelphia, 2002.

[129] H. P. Hong and C. T. Pan, "Rank-revealing QR factorization and SVD." Math. Comp. **58**:213–232, 1992.

[130] P. J. Huber and E. M. Ronchetti, *Robust Statistics.* Second edition, J. Wiley, NJ, 2009.

[131] R. Horst, P. M. Pardalos and N. Van Thoai, *Introduction to Global Optimization.* Second edition, Springer, New York, 2000.

[132] N. J. Horton and S. R. Lipsitz, "Multiple imputation in practice: Comparison of software packages for regression models with missing variables." The American Statistician **55**, 2011.

[133] I. C. F. Ipsen and C. D. Meyer, "The idea behind Krylov methods." Amer. Math. Monthly **105**:889–899, 1998.

[134] R. A. Johnson, *Miller & Freund's Probability and Statistics for Engineers.* Pearson Prentice Hall, Upper Saddle River, NJ, 2005.

[135] P. Jones, *Data Tables of Global and Hemispheric Temperature Anomalies*
http://cdiac.esd.ornl.gov/trends/temp/jonescru/data.html.

[136] T. L. Jordan, "Experiments on error growth associated with some linear least squares procedures." Math. Comp. **22**:579–588, 1968.

[137] D. L. Jupp, "Approximation to data by splines with free knots." SIAM J. Numer. Anal. **15**:328, 1978.

[138] D. L. Jupp and K. Vozoff, "Stable iterative methods for the inversion of geophysical data." J. R. Astr. Soc. **42**:957–976, 1975.

[139] W. Karush, "Minima of functions of several variables with inequalities as side constraints." M. Sc. Dissertation. Department of Mathematics, University of Chicago, Chicago, Illinois, 1939.

[140] L. Kaufman, "A variable projection method for solving nonlinear least squares problems." BIT **15**:49–57, 1975.

[141] L. Kaufman and G. Sylvester, "Separable nonlinear least squares with multiple right-hand sides." SIAM J. Matrix Anal. and Appl. **13**:68–89, 1992.

[142] L. Kaufman and V. Pereyra, "A method for separable nonlinear least squares problems with separable equality constraints." SIAM J. Numer. Anal. **15**:12–20, 1978.

[143] H. B. Keller and V. Pereyra, *Finite Difference Solution of Two-Point BVP.* Preprint 69, Department of Mathematics, University of Southern California, 1976.

[144] M. E. Kilmer and D. P. O'Leary, "Choosing regularization parameters in iterative methods for ill-posed problems." SIAM. J. Matrix Anal. Appl. **22**:1204–1221, 2001.

[145] H. Kim, G. H. Golub and H. Park, "Missing value estimation for DNA microarray expression data: Least squares imputation." In Proceedings of CSB'2004, Stanford, CA, pp. 572–573, 2004.

[146] S. Kotz, N. L. Johnson and A. B. Read, *Encyclopedia of Statistical Sciences.* **6**. Wiley-Interscience, New York, 1985.

[147] H. W. Kuhn and A. W. Tucker, *Nonlinear Programming.* Proceedings of 2nd Berkeley Symposium, 481–492, University of California Press, Berkeley, 1951.

[148] *Lancelot: Nonlinear Programming Code* http://www.numerical.rl.ac.uk/lancelot/

[149] K. Lange, *Numerical Analysis for Statisticians.* Second edition, Springer, New York, 2010.

[150] C. Lawson and R. Hanson, *Solving Least Squares Problems.* Prentice Hall, Englewood Cliffs, NJ, 1974.

[151] K. Levenberg, "A method for the solution of certain nonlinear problems in least squares." Quart. J. App. Math. **2**:164–168, 1948.

[152] R. J. A. Little and D. B. Rubin, *Statistical Analysis with Missing Data.* Second edition. J. Wiley, Hoboken, NJ, 2002.

[153] K. Madsen and H. B. Nielsen, "A finite smoothing algorithm for linear ℓ_1 estimation." SIAM J. Optimiz. **3**:223–235, 1993.

[154] K. Madsen and H. B. Nielsen, *Introduction to Optimization and Data Fitting.* Lecture notes, Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, 2008.

[155] I. Markovsky and S. Van Huffel, "Overview of total least-squares methods." Signal Processsing **87**:2283–2302, 2007.

[156] O. Mate, "Missing value problem." Master's Thesis, Stanford University, Stanford, CA, 2007.

[157] Matrix Market. http://math.nist.gov/MatrixMarket

[158] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra.* SIAM, Philadelphia, 2000.

[159] J.-C. Miellou, "Iterations chaotiques a retards." C. R. Acad. Sci. Paris **278**:957–960, 1974.

[160] J.-C. Miellou, "Algorithmes de relaxation chaotique a retards." RAIRO **R1**:55–82, 1975.

[161] K. M. Miettinen, *Nonlinear Multiobjective Optimization*. Kluwers Academic, Boston, 1999.

[162] L. Miranian and M. Gu, "Strong rank revealing LU factorizations." Lin. Alg. Appl. **367**:1–16, 2003.

[163] Software for multiple imputation. http://www.stat.psu.edu/~jls/misoftwa.html

[164] M. Mohlenkamp and M. C. Pereyra, *Wavelets, Their Friends, and What They Can Do For You*. EMS Lecture Series in Mathematics, European Mathematical Society, Zurich, Switzerland, 2008.

[165] J. Moré and S. J. Wright, *Optimization Software Guide*. SIAM, Philadelphia, 1993.

[166] A. Nedic, D. P. Bertsekas and V. S. Borkar, "Distributed asynchronous incremental subgradient methods." Studies in Computational Mathematics **8**:381–407, 2001.

[167] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Methods in Convex Programming*. Studies in Applied Mathematics 13. SIAM, Philadelphia, 1994.

[168] L. Ngia, "System modeling using basis functions and applications to echo cancellation." Ph.D. Thesis, Chalmers Institute of Technology, Sweden, Goteborg, 2000.

[169] L. Ngia, *Separable Nonlinear Least Squares Methods for On-Line Estimation of Neural Nets Hammerstein Models*. Department of Signals and Systems, Chalmers Institute of Technology, Sweden, 2001.

[170] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, New York, second edition, 2006.

[171] D. P. O'Leary, "Robust regression computation using iteratively reweighted least squares." SIAM J. Matrix. Anal. Appl. **22**:466–480, 1990.

[172] D. P. O'Leary and B. W. Rust, "Variable projection for nonlinear least squares problems." Submitted for publication in Computational Optimization and Applications (2011). Also at: http://www.cs.umd.edu/users/oleary/software/varpro.pdf

[173] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York, 1970.

[174] M. R. Osborne, "Some special nonlinear least squares problems." SIAM J. Numer. Anal. **12**:571–592, 1975.

[175] M. R. Osborne and G. K. Smyth, "A modified Prony algorithm for fitting functions defined by difference equations." SIAM J. Sci. Comp. **12**:362–382, 1991.

[176] M. R. Osborne and G. K. Smyth, "A modified Prony algorithm for exponential function fitting." SIAM J. Sci. Comp. **16**:119–138, 1995.

[177] M. R. Osborne, "Separable least squares, variable projections, and the Gauss-Newton algorithm." ETNA **28**:1–15, 2007.

[178] A. Ostrowski, "Determinanten mit ueberwiegender Hauptdiagonale and die absolute Konvergenz von linearen Iterationsprozessen." Comm. Math. Helv. **30**:175–210, 1955.

[179] C. C. Paige and M. A. Saunders, "LSQR: An algorithm for sparse linear equations and sparse least squares." ACM TOMS **8**:43–71, 1982.

[180] C. C. Paige and Z. Strakos, "Core problems in linear algebraic systems." SIAM J. Matrix Anal. Appl. **27**:861–875, 2006.

[181] R. Parisi, E. D. Di Claudio, G. Orlandi and B. D. Rao, "A generalized learning paradigm exploiting the structure of feedforward neural networks." IEEE Transactions on Neural Networks **7**:1450–1460, 1996.

[182] Y. C. Pati and P. S. Krishnaprasad, "Analysis and synthesis of feedforward neural networks using discrete affine wavelet transformations." IEEE Trans. Neural Networks **4**:73–85, 1993.

[183] V. Pereyra, "Accelerating the convergence of discretization algorithms." SIAM J. Numer. Anal. **4**:508–533, 1967.

[184] V. Pereyra, "Iterative methods for solving nonlinear least squares problems." SIAM J. Numer. Anal. **4**:27–36, 1967.

[185] V. Pereyra, "Stability of general systems of linear equations." Aeq. Math. **2**:194–206, 1969.

[186] V. Pereyra, "Stabilizing linear least squares problems." Proc. IFIP, Suppl. **68**:119–121, 1969.

[187] V. Pereyra, "Modeling, ray tracing, and block nonlinear travel-time inversion in 3D." Pure and App. Geoph. **48**:345–386, 1995.

[188] V. Pereyra, "Asynchronous distributed solution of large scale nonlinear inversion problems." J. App. Numer. Math. **30**:31–40, 1999.

[189] V. Pereyra, "Ray tracing methods for inverse problems." Invited topical review. Inverse Problems **16:**R1–R35, 2000.

[190] V. Pereyra, "Fast computation of equispaced Pareto manifolds and Pareto fronts for multiobjective optimization problems." Math. Comp. in Simulation **79**:1935–1947, 2009.

[191] V. Pereyra, M. Koshy and J. Meza, "Asynchronous global optimization techniques for medium and large inversion problems." SEG Annual Meeting Extended Abstracts **65**:1091–1094, 1995.

[192] V. Pereyra and P. Reynolds, "Application of optimization techniques to finite element analysis of piezocomposite devices." IEEE Ultrasonics Symposium, Montreal, CANADA, 2004.

[193] V. Pereyra and J. B. Rosen, *Computation of the Pseudoinverse of a Matrix of Unknown Rank.* Report CS13, 39 pp. Computer Science Department, Stanford University, Stanford, CA, 1964.

[194] V. Pereyra, M. Saunders and J. Castillo, *Equispaced Pareto Front Construction for Constrained Biobjective Optimization.* SOL Report-2010-1, Stanford University, Stanford, CA, 2010. Also CSRCR2009-05, San Diego State University, 2009. In Press, Mathematical and Computer Modelling, 2012.

[195] V. Pereyra and G. Scherer, "Efficient computer manipulation of tensor products with applications in multidimensional approximation." Math. Comp. **27**:595–605, 1973.

[196] V. Pereyra and G. Scherer, "Least squares scattered data fitting by truncated SVD." Applied Numer. Math. **40**:73–86, 2002.

[197] V. Pereyra and G. Scherer, "Large scale least squares data fitting." Applied Numer. Math. **44**:225–239, 2002.

[198] V. Pereyra and G. Scherer, "Exponential data fitting." In *Exponential Data Fitting and Its Applications*, Ed. V. Pereyra and G. Scherer. Bentham Books, Oak Park, IL, 2010.

[199] V. Pereyra, G. Scherer and F. Wong, "Variable projections neural network training." Mathematics and Computers in Simulation **73**:231–243, 2006.

[200] V. Pereyra, G. Wojcik, D. Powell, C. Purcell and L. Carcione, "Folded shell projectors and virtual optimization." US Navy Workshop on Acoustic Transduction Materials and Devices, Baltimore, 2001.

[201] G. Peters and J. H. Wilkinson, "The least squares problem and pseudo-inverses." The Computer Journal **13**:309–316, 1969.

[202] M. J. D. Powell, *Approximation Theory and Methods.* Cambridge University Press, New York, 1981.

[203] L. Prechelt, *Proben 1-A Set of Benchmark Neural Network Problems and Benchmarking Rules.* Technical Report 21, Fakultaet fuer Informatik, Universitaet Karlsruhe, 1994.

[204] Baron Gaspard Riche de Prony, "Essai experimental et analytique: sur les lois de la dilatabilite de fluides elastique et sur celles de la force expansive de la vapeur de l'alkool, a differentes temperatures." J. Ecole Polyt. **1**:24–76, 1795.

[205] *PZFLEX: Weidlinger Associates Inc. Finite Element Code to Simulate Piezoelectric Phenomena.* http://www.wai.com, 2006.

[206] L. Reichel, "Fast QR decomposition of Vandermonde-like matrices and polynomial least squares approximations." SIAM J. Matrix Anal. Appl. **12**:552–564, 1991.

[207] R. A. Renaut and H. Guo, "Efficient algorithms for solution of regularized total least squares." SIAM J. Matrix Anal. Appl. **26**:457–476, 2005.

[208] J. R. Rice, "Experiments on Gram-Schmidt orthogonalization." Math. Comp. **20**:325–328, 1966.

[209] J. B. Rosen, "Minimum and basic solutions to singular systems." J. SIAM **12**:156–162, 1964.

[210] J. B. Rosen and J. Kreuser, "A gradient projection algorithm for nonlinear constraints." In Numerical Methods for Non-Linear Optimization. Ed. F. A. Lootsma, Academic Press, New York, pp. 297–300, 1972.

[211] J. Rosenfeld, *A Case Study on Programming for Parallel Processors.* Research Report RC-1864, IBM Watson Research Center, Yorktown Heights, New York, 1967.

[212] Å. Ruhe and P-Å. Wedin, "Algorithms for separable non-linear least squares problems." SIAM Rev. **22**:318–337, 1980.

[213] B. W. Rust, "Fitting nature's basis functions." Parts I-IV, Computing in Sci. and Eng. 2001–03.

[214] B. W. Rust, http://math.nist.gov/~BRust/Gallery.html, 2011.

[215] B. W. Rust, *Truncating the Singular Value Decomposition for Ill-Posed Problems.* Tech. Report NISTIR 6131, Mathematics and Computer Sciences Division, National Institute of Standards and Technology, 1998.

[216] B. W. Rust and D. P. O'Leary, "Residual periodograms for choosing regularization parameters for ill-posed problems." Inverse Problems **24**, 2008.

[217] S. Saarinen, R. Bramley and G. Cybenko, "Ill-conditioning in neural network training problems." SIAM J. Sci. Stat. Comput. **14**:693–714, 1993.

[218] S. A. Savari and D. P. Bertsekas, "Finite termination of asynchronous iterative algorithms." Parallel Comp. **22**:39–56, 1996.

[219] J. A. Scales, A. Gersztenkorn and S. Treitel, "Fast l_p solution of large, sparse, linear systems: application to seismic travel time tomography." J. Comp. Physics **75**:314–333, 1988.

[220] J. L. Schafer, *Analysis of Incomplete Multivariate Data.* Chapman & Hall, London, 1997.

[221] S. Schechter, "Relaxation methods for linear equations." Comm. Pure Appl. Math. **12**:313–335, 1959.

[222] M. E. Schlesinger and N. Ramankutty, "An oscillation in the global climate system of period 65–70 years." Nature **367**:723–726, 1994.

[223] G. A. F. Seber and C. J. Wild, *Nonlinear Regression.* J. Wiley, New York, 1989.

[224] R. Sheriff and L. Geldart, *Exploration Seismology.* Cambridge University Press, second edition, Cambridge, 1995.

[225] D. Sima, S. Van Huffel and G. Golub, "Regularized total least squares based on quadratic eigenvalue problem solvers." BIT **44**:793–812, 2004.

[226] Spline2. http://www.structureandchange.3me.tudelft.nl/.

[227] J. Sjöberg and M. Viberg, "Separable non-linear least squares minimization–possible improvements for neural net fitting." IEEE Workshop in Neural Networks for Signal Processing, Amelia Island Plantation, FL, 1997.

[228] N. Srivastava, R. Suaya, V. Pereyra and K. Banerjee, "Accurate calculations of the high-frequency impedance matrix for VLSI interconnects and inductors above a multi-layer substrate: A VARPRO success story." In *Exponential Data Fitting and Its Applications*, Eds. V. Pereyra and G. Scherer. Bentham Books, Oak Park, IL, 2010.

[229] T. Steihaug and Y. Yalcinkaya, "Asynchronous methods in least squares: An example of deteriorating convergence." Proc. 15 IMACS World Congress on Scientific Computation, Modeling and Applied Mathematics, Berlin, Germany, 1997.

[230] G. W. Stewart, "Rank degeneracy." SIAM J. Sci. Stat. Comput. **5**:403–413, 1984.

[231] G. W. Stewart, "On the invariance of perturbed null vectors under column scaling." Numer. Math. **44**:61–65, 1984.

[232] G. W. Stewart, *Matrix Algorithms. Volume I: Basic Decompositions.* SIAM, Philadelphia, 1998.

[233] T. Strutz, *Data Fitting and Uncertainty.* Vieweg+Teubner Verlag, Wiesbaden, 2011.

[234] B. J. Thijsse and B. Rust, "Freestyle data fitting and global temperatures." Computing in Sci. and Eng. pp. 49–59, 2008.

[235] R. Tibshirani, "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society. Series B (Methodological) **58**:267–288, 1996.

[236] A. N. Tikhonov, "On the stability of inverse problems." Dokl. Akad. Nauk SSSR **39**:195–198, 1943.

[237] L. N. Trefethen and D. Bau, *Numerical Linear Algebra.* SIAM, Philadelphia, 1997.

[238] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein and R. B. Altman, "Missing value estimation methods for DNA microarrays." Bioinformatics **17**:520–525, 2001.

[239] S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem. Computational Aspects and Analysis,* SIAM, Philadelphia, 1991.

[240] E. van den Berg and M. P. Friedlander, "Probing the Pareto frontier for basis pursuit solutions." SIAM J. Sci. Comp. **31**:890–912, 2008.

[241] E. van den Berg and M. P. Friedlander, "Sparse optimization with least-squares constraints." SIAM J. Optim. **21**:1201–1229, 2011.

[242] A. van den Bos, *Parameter Estimation for Scientists and Engineers.* Wiley-Interscience, Hoboken, NJ, 2007.

[243] A. van der Sluis, "Condition numbers and equilibration of matrices." Numer. Math. **14**:14–23, 1969.

[244] J. W. van der Veen, R. de Beer, P. R. Luyten and D. Van Ormondt, "Accurate quantification of in vivo 31P NMR signals using the variable projection method and prior knowledge." Magn. Reson. Med. **6**:92–98, 1988.

[245] L. Vanhamme, A. van den Boogaart and S. Van Huffel, "Improved method for accurate and efficient quantification of MRS data with use of prior knowledge." J. Magn. Reson. **129**:35–43, 1997.

[246] L. Vanhamme, S. Van Huffel, P. Van Hecke and D. van Ormondt, "Time-domain quantification of series of biomedical magnetic resonance spectroscopy signals." J. Magn. Reson. **140**:120–130, 1999.

[247] L. Vanhamme, T. Sundin, P. Van Hecke, S. Van Huffel and R. Pintelon, "Frequency-selective quantification of biomedical magnetic resonance spectroscopy data." J. Magn. Reson. **143**:1–16, 2000.

[248] S. Van Huffel, 'Partial singular value decomposition algorithm." J. Comp. Appl. Math. **33**:105–112, 1990.

[249] B. Walden, R. Karlsson and J.-G. Sun, "Optimal backward perturbation bounds for the linear least squares problem." Numer. Linear Algebra Appl. **2**:271–286, 2005.

[250] I. Wasito and B. Mirkin, "Nearest neighbors in least squares data imputation algorithms with different missing patterns." Comp. Stat. & Data Analysis **50**: 926–949, 2006.

[251] D. S. Watkins, *Fundamentals of Matrix Computations.* J. Wiley, New York, 2002.

[252] K. Weigl and M. Berthod, *Neural Networks as Dynamical Bases in Function Space.* Report 2124, INRIA, Programe Robotique, Image et Vision, Sophia-Antipolis, France, 1993.

[253] K. Weigl, G. Giraudon and M. Berthod, *Application of Projection Learning to the Detection of Urban Areas in SPOT Satellite Images.* Report #2143, INRIA, Programe Robotique, Image et Vision, Sophia-Antipolis, France, 1993.

[254] K. Weigl and M. Berthod, "Projection learning: Alternative approach to the computation of the projection." Proceedings of the European Symposium on Artificial Neural Networks pp. 19–24, Brussels, Belgium, 1994.

[255] J. H. Wilkinson, *Rounding Errors in Algebraic Processes.* Prentice-Hall, Englewood Cliffs, NJ, 1963. Reprint Dover, New York, 1994.

[256] H. Wold and E. Lyttkens, "Nonlinear iterative partial least squares (NIPALS) estimation procedures." Bull. ISI **43**:29–51, 1969.

[257] S. Wold, A. Ruhe, H. Wold and W. J. Dunn, "The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses." SIAM J. Sci. Stat. Comput. **5**:735–743, 1984.

[258] R. Wolke and H. Schwetlick, "Iteratively reweighted least squares: Algorithms, convergence analysis, and numerical comparisons." SIAM J. Sci. Stat. Comput. **9**:907–921, 1988.

[259] S. J. Wright, *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, 1997.

[260] S. J. Wright, J. N. Holt, "Algorithms for non-linear least squares with linear inequality constraints." SIAM J. Sci. and Stat. Comput. **6**:1033–1048, 1985.

[261] P. Zadunaisky and V. Pereyra, "On the convergence and precision of a process of successive differential corrections." Proc. IFIPS **65**, 1965.

[262] H. Zha, "Singular values of a classical matrix." The American Mathematical Monthly **104**:172–173, 1997.

[263] Netlib Repository at UTK and ORNL. http://www.netlib.org.

[264] NEOS Wiki. http://www.neos-guide.org.

[265] NIST/SEMATECH e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/strd/nls/nls_info.shtml, 2011.

Index

Activation function, 232
Active constraint, 127, 156
Aerodynamic modeling, 242
Annual temperature anomalies, 213
Asynchronous
    block Gauss-Seidel, 190
    iterative methods, 189
    method, 117
Autocorrelation, 15, 216, 218
    test, 14

B-spline, 260
B-splines representation, 203
Back-scattering, 258
Backward stable, 88, 265
Basic solution, 41, 42, 91, 94, 97, 145
Bidiagonalization, 99–102, 112
    method
        Golub-Kahan, 111
        LSQR, 111
Block
    Gauss-Seidel, 117, 224
    Jacobi, 117
    methods, 117, 186
    nonlinear Gauss-Seidel, 186, 259
    nonlinear Jacobi, 186
Bolzano-Weierstrass theorem, 156
Bound constraints, 126, 127

Cauchy-Schwartz inequality, 267
CFL condition, 254
CGLS solution convergence, 116
Chaotic relaxation, 117, 188
Chebyshev
    acceleration, 107
    method, 107
    norm, 267
Chi-square distribution, 278
Cholesky factor, 34, 66, 114
Coefficient of determination, 38, 278
    adjusted, 38, 216
Complex exponentials, 184
Compressed sensing, 4, 41, 91, 121, 144
Computed tomography (CT), 110
Condition number, 55, 268
    estimation, 88
Conjugate gradient method, 109
Control vertices, 205
Covariance, 31, 277
    matrix, 31
        approximation, 155
        method, 81
Cubic splines, 203

Data approximation, 2
Data fitting, 4
    problem, 1
Data imputation, 131
Derivative free methods, 183
Descent direction, 153, 272
Direct methods, 65, 91
Distribution function, 276

Eckart-Young-Mirski theorem, 53
Elastic waves, 258
Euclidean norm, 267
Expectation, 276

F-distribution, 279

Feasible region, 156
Finite-element simulation, 238
First-order necessary condition, 151
Fitting model, 4, 13, 16
Flop, 264
Fréchet derivative, 50
Frobenius norm, 268

Gaussian
  density function, 11
  distribution, 10, 11
  errors, 12
  model, 147, 155
  probability function, 278
Gauss-Newton
  damped, 167
  direction, 166
  inexact, 170
  method, 163
Generalized cross-validation, 197
Genetic algorithm, 248
Geological
  medium, 259
  surface modeling, 221
Geophone receiver array, 259
Givens rotations, 71
  fast, 72
Globally convergent algorithm, 271
Global minimum, 271
Global optimization, 177, 241
Gradient vector, 272
Grammian, 28
  matrix, 65
Gram-Schmidt
  modified, 77
  orthogonalization, 70, 75
Ground boom signature, 242

Hessian, 28, 152, 163, 165, 272
Householder transformations, 71, 122
Hybrid methods, 176
Hypothesis testing, 280

IEEE standard shapes, 252
Ill-conditioned, 191, 207, 223
  Jacobian, 168, 176
Ill-conditioning, 191
Inexact methods, 186
Interlacing property, 80, 123, 141, 270
Interpolation, 3
Inverse problem, 129
Iterative
  methods, 105, 107, 108, 119
  process, 117
  refinement, 85

Jacobian, 272

Kronecker product, 209, 210
Krylov
  iterations, 194
  process, 111
  subspace, 108
    methods, 225

Lagrange multipliers, 158, 273
Lagrangian, 273
Laplace
  density function, 11
  distribution, 11
  errors, 12
Large-scale problems, 105
Least squares
  data fitting problem, 6
  fit, 5, 7, 9
  overdetermined, 68
  recursive, 81
Least squares problems
  condition number, 47
  large
    linear, 117
  linear, 25
    constrained, 121
  minimum-norm solution, 93
  modifying, 80
  rank deficient, 39
Least squares solution

full-rank problem, 34
Level curves, 63
Level of significance, 280
Level set, 153
Levenberg-Marquardt
algorithm, 170
asynchronous BGS iteration, 190
direction, 171
implementation, 250
method, 170
programs, 178
step, 171
Linear constraints, 121
equality, 121
inequality, 125
Linear data fitting, 4
Linearized sensitivity analysis, 255
Linear least squares applications, 203
Linear prediction, 41
Lipschitz continuity, 273
Local
minimizer, 156, 271
minimum, 151
Locally convergent algorithm, 271
Log-normal model, 12
LU factorization, 68
Peters-Wilkinson, 93

Machine epsilon, 263
Matrix norm, 267
Maximum likelihood principle, 6, 9
Minimal solution, 145
Model basis functions, 4
Model validation, 217
Monte Carlo, 176
Moore-Penrose
conditions, 53
generalized inverse, 182
Multiobjective optimization, 160
Multiple initial points, 176
random, 163
Multiple right-hand sides, 184

Neural networks, 231
training algorithm, 231
Newton's method, 163
NIPALS, 181
NMR, 7
nuclear magnetic resonance, 2, 248
spectroscopy, 150
problem, 29
Non-dominated solutions, 248
Nonlinear data fitting, 148
Nonlinear least squares, 147, 148
ill-conditioned, 201
large scale, 186
separable, 158, 231, 234, 236
unconstrained, 150
Non-stationary methods
Krylov methods, 108
Normal distribution, 14
Normal equations, 28, 65
Normalized cumulative periodograms, 16
Numerically rank-deficient problems, 192
Numerical rank, 92

Operator norm, 267
Optimality conditions, 151
constrained problems, 156
Order of fit, 4
Orthogonal factorization
complete, 43
Orthogonal matrices, 269
Orthogonal projection, 269
derivative, 49

Parameter estimation, 1
Pareto
equilibrium point, 160
front, 161, 248
equispaced representation, 161
optimal, 160
Pearson correlation coefficient, 277
Permutation matrix, 269
Perturbation analysis, 58

Piezoelectric transducer, 251
Poisson data, 11
PRAXIS, 183, 252
Preconditioning, 114
Probability density function, 275
Probability function, 275
Probability of an event, 275
Projection matrix, 48
Proper splitting, 107
Proxies, 238
Pseudoinverse, 47
Pure-data function, 4, 10

QR factorization, 33, 70, 94
    economical, 34
    full, 82
    pivoted, 39, 96
    rank-revealing, 95
Quadratic constraints, 129

Randomness test, 14
Random signs, 14
Random variable, 275
    continuous, 275
    discrete, 275
Rank deficiency, 191
Rank-deficient problems, 91
Rational model, 148
Regularization, 192, 241
Residual analysis, 16
Response surfaces, 238, 244
Root mean square, 277

Sample space, 275
Second-order
    sufficient conditions, 151
Secular equation, 130
Seismic
    ray tracing, 259
    survey, 258
Sensitivity analysis, 59, 127
Sherman-Morrison-Woodbury
    formula, 270
Sigmoid activation functions, 244
Signal to noise ratio, 278
Singular value, 50, 110
    normalized, 92
Singular value decomposition, 47, 50
    economical, 50
    generalized, 54, 122, 129
    partial, 140
Singular vectors, 50
Stability, 88
Standard deviation, 185, 277
Standard error, 37
Stationary methods, 107
Stationary point, 272
Statistically independent, 277
Steepest descent, 153
Step length, 177
Stopping criteria, 114
Sum-of-absolute-values, 5, 11
Surrogates, 238
SVD computations, 101
Systems of linear equations
    parallel iterative solution, 188

Taylor's theorem, 272
Tensor product, 209
    bivariate splines, 208
    cubic B-splines, 222
    data, 224
    fitting, 224
Tikhonov regularization, 118, 163
Toeplitz matrix, 41, 88
Total least squares, 136
Truncated SVD, 118

Uniformly monotone operator, 273
Unitary matrix, 34
Unit round-off, 263
UTV decomposition, 98

Vandermonde matrix, 57
Variable projection, 231, 235, 249
    algorithm, 181, 183
    principle, 181
Variance, 276

VARP2, 184
VARPRO program, 183, 236
Vector
    contracting, 273
    Lipschitz, 273
    norms, 267

Weighted residuals, 6
White noise, 5, 14
    test, 14, 15
    vector, 278